Talleres Ciencia de Datos con R. Análisis de Datos Bloque1 Lecciones 1 a 16

Ricardo Alberich

13 de septiembre de 2018

Contents

1 Tareas Sección 3: Lecciones 14 a 27.

1.1 Tarea 1: Visualización de datos con ggplot2. Sección 3: Lecciones 14 a 16.

1.1.1 Pregunta 1.1.

Si ejecutas `ggplot(data = mpg)`, ¿Qué observas?

1.1.1.1 Solución

Inicializa el gráfico vacío a las espera de otras especificaciones de lo que se tiene que dibujar.

Podemos ver un recuadro vacío, lo que observamos es la base para un gráfico.

```
ggplot(data = mpg)
```


1.1.2 Pregunta 1.2.

Indica el número de filas que tiene el data frame `mpg`. ¿Qué significa cada fila?

1.1.2.1 Solución

Usando el comando nrow() sabemos que tiene 234 filas y observando con el comando View() se deduce que cada fila conforma las especificaciones de cada vehículo.

```
nrow(mpg)
```

```
## [1] 234
```

1.1.3 Pregunta 1.3.

Indica el número de columnas que tiene el data frame mpg. ¿Qué significa cada columna?

1.1.3.1 Solución

De la misma manera que con las filas usando el comando ncol() sabemos que el número de columnas es 11 y consultando la información auxiliar con el comando ? entendemos que cada columna refleja una característica.

"manufacturer": marca

"model name": Nombre del modelo

“displ”: ciclindrada en litros

“year”: año de producción

“cyl”: número de cilindros

“trans”: tipo de transmisión

“drw”: tipo de tracción (f: delantera,r: trasera, 4: 4 ruedas)

“cty”: millas por galón en ciudad

“hwy”: millas por galón en autopista

“fl”: tipo de combustible

“class”: tipo de coche

```
ncol(mpg)
```

```
## [1] 11
```

1.1.4 Pregunta 1.4.

Observa la variable drv del data frame. ¿Qué describe? Recuerda que puedes usar la instrucción ?mpg para consultarlo directamente en R.

1.1.4.1 Solución

Como ha sido comentado anteriormente “drv” indica el tipo de tracción del coche.

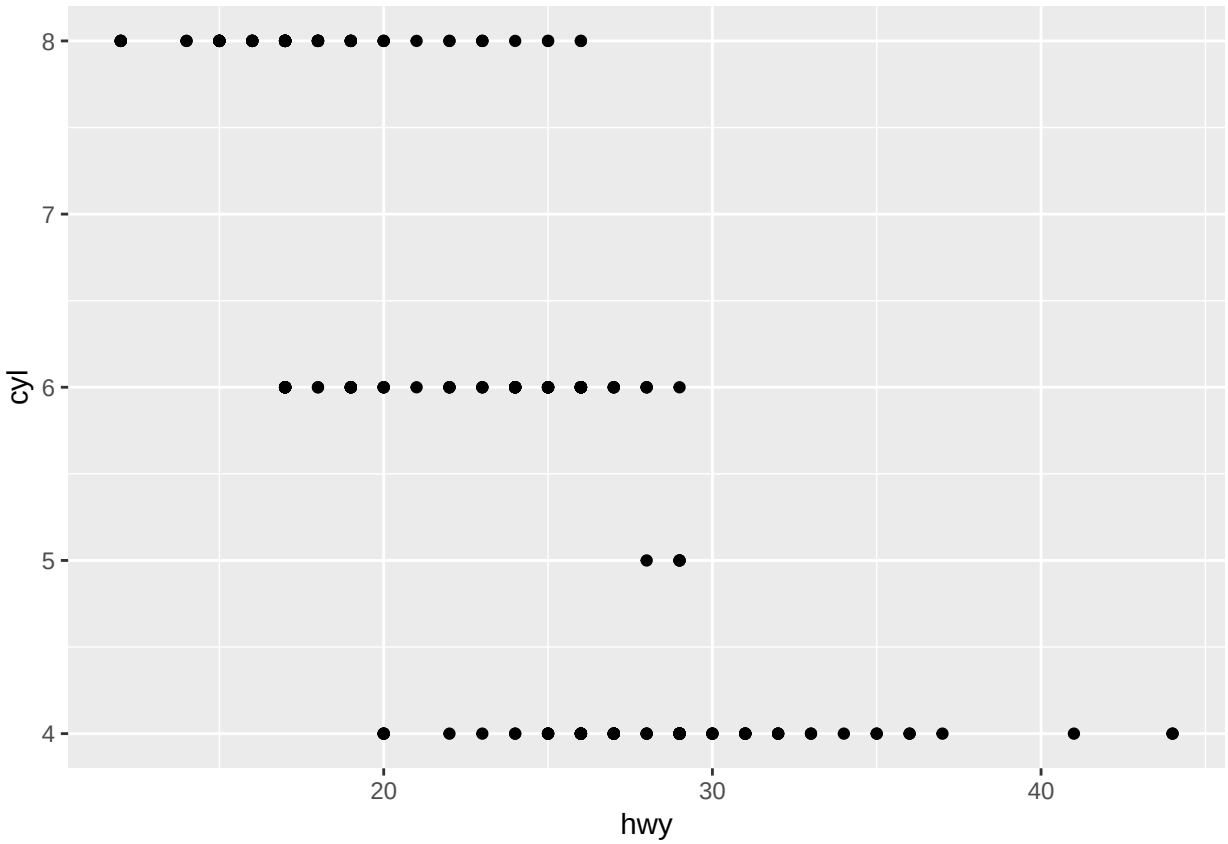
1.1.5 Pregunta 1.5.

Realiza un scatterplot de la variable hwy vs cyl. ¿Qué observas?

1.1.5.1 Solución

Podemos ver una gráfica en la que se comparara el número de cilindros y la eficiencia en millas por galón. A primera vista parece indicar que los coches con menos cilindros son más eficientes.

```
ggplot(data=mpg)+ geom_point(mapping= aes(x=hwy , y=cyl))
```



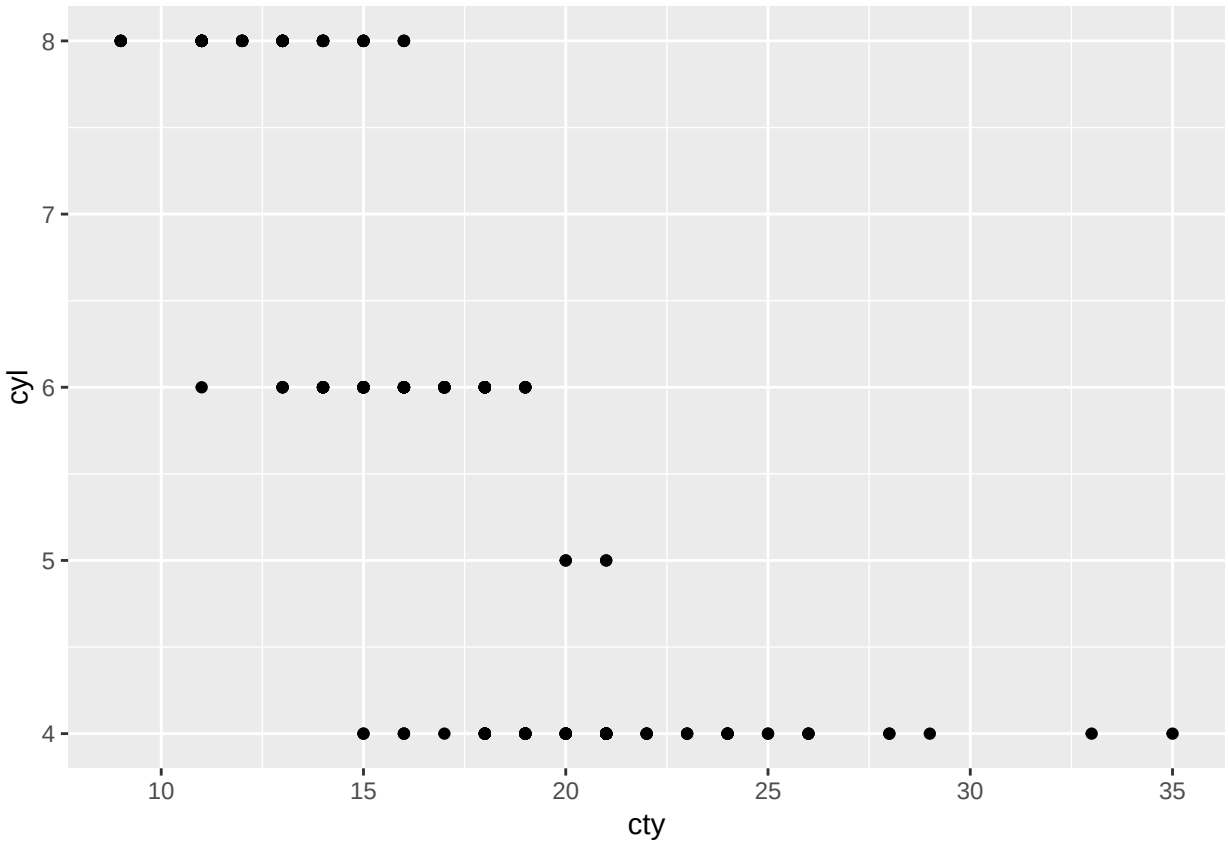
1.1.6 Pregunta 1.6.

Realiza un scatterplot de la variable cty vs cyl. ¿Qué observas?

1.1.6.1 Solución

Vemos una comparativa entre el número de millas por galón en ciudad y el número de cilindros. A primera vista parece indicar que los coches con menos cilindros son mas eficientes

```
ggplot(data=mpg)+ geom_point(mapping= aes(x= cty , y=cyl))
```



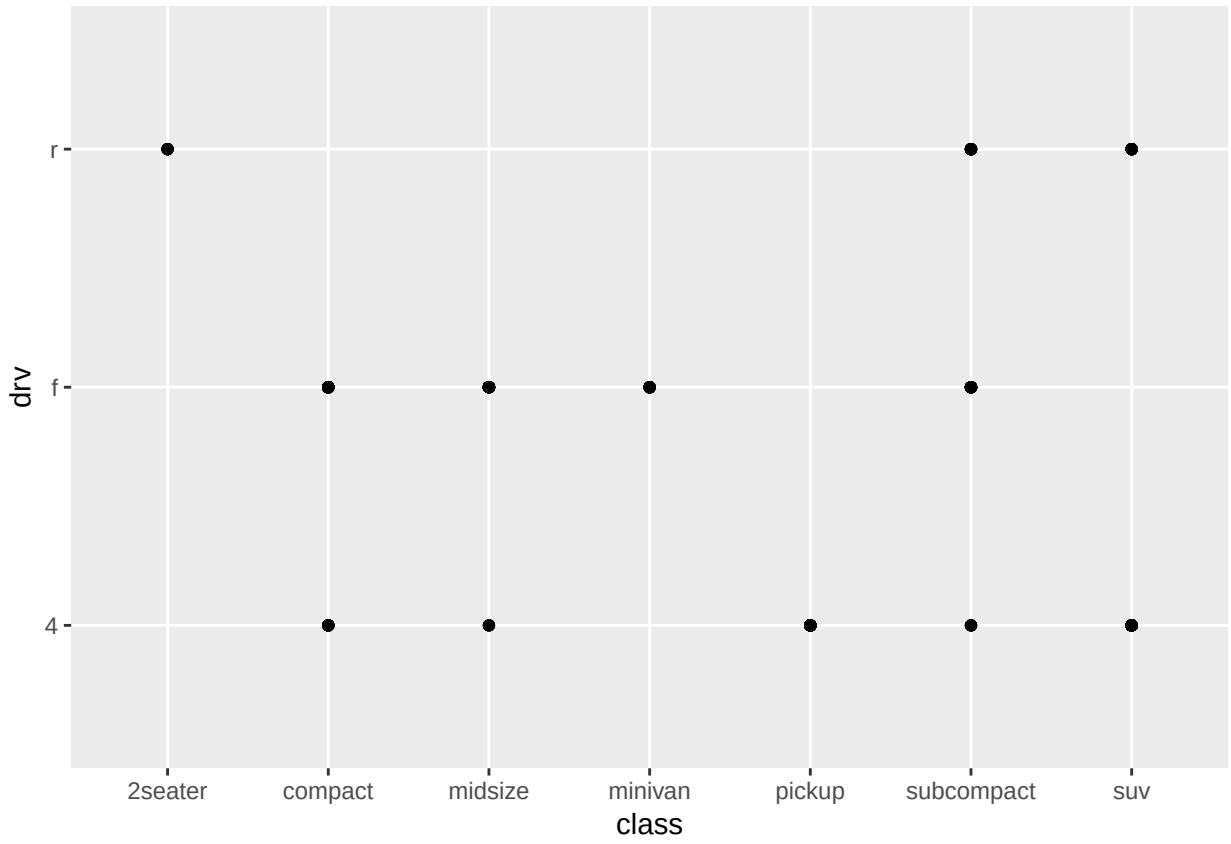
1.1.7 Pregunta 1.7.

Realiza un scatterplot de la variable class vs drv. ¿Qué observas? ¿Es útil este diagrama? ¿Por qué?

1.1.7.1 Solución

Observamos una comparativa entre la clase del coche y si tracción. Este diagrama no es útil ya que compara dos variables sin interés ya que nuestro objetivo es estudiar la eficiencia. En general no aporta datos relevantes.

```
ggplot(data=mpg)+ geom_point(mapping= aes(x=class , y=drv))
```



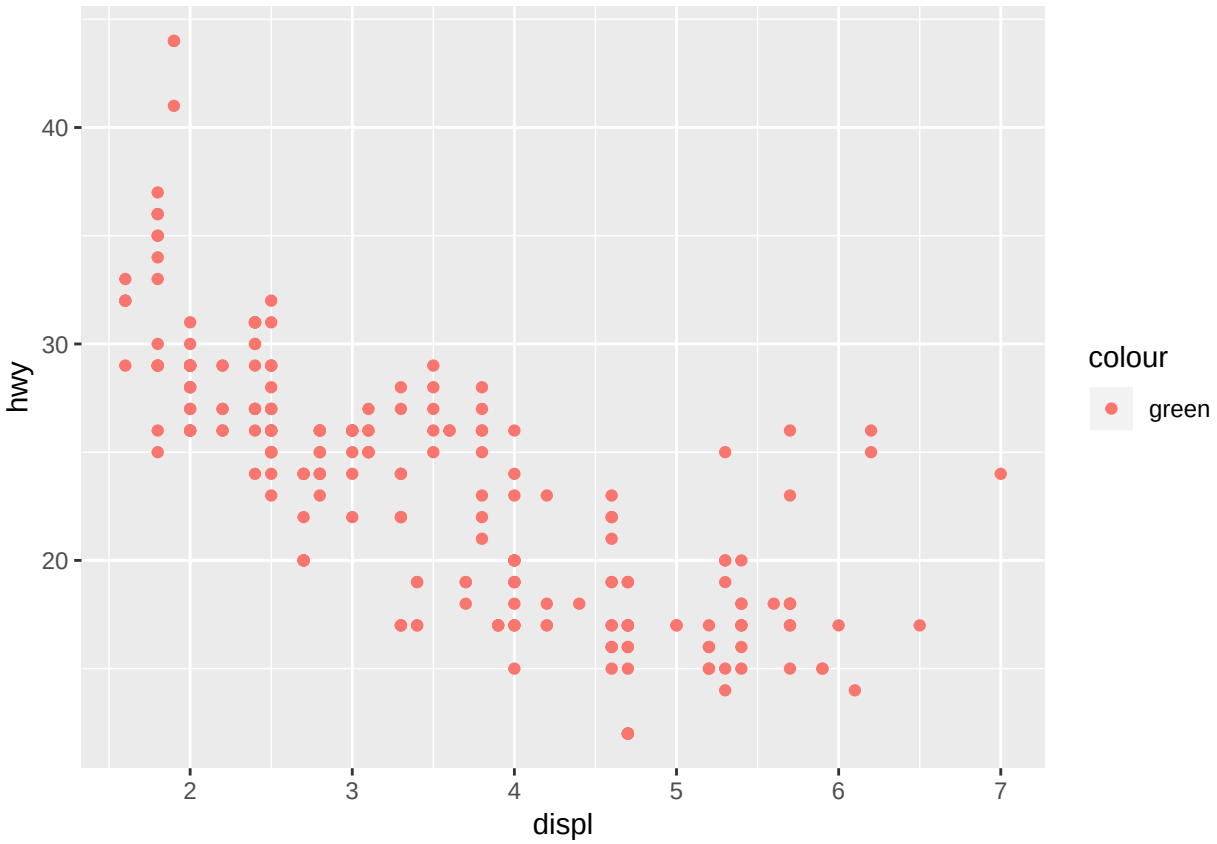
1.2 Tarea 2: Estéticas ggplot. Sección 3: Lección 17.

1.2.1 Pregunta 2.1.

Toma el siguiente fragmento de código y di qué está mal. ¿Por qué no aparecen pintados los puntos de color verde?

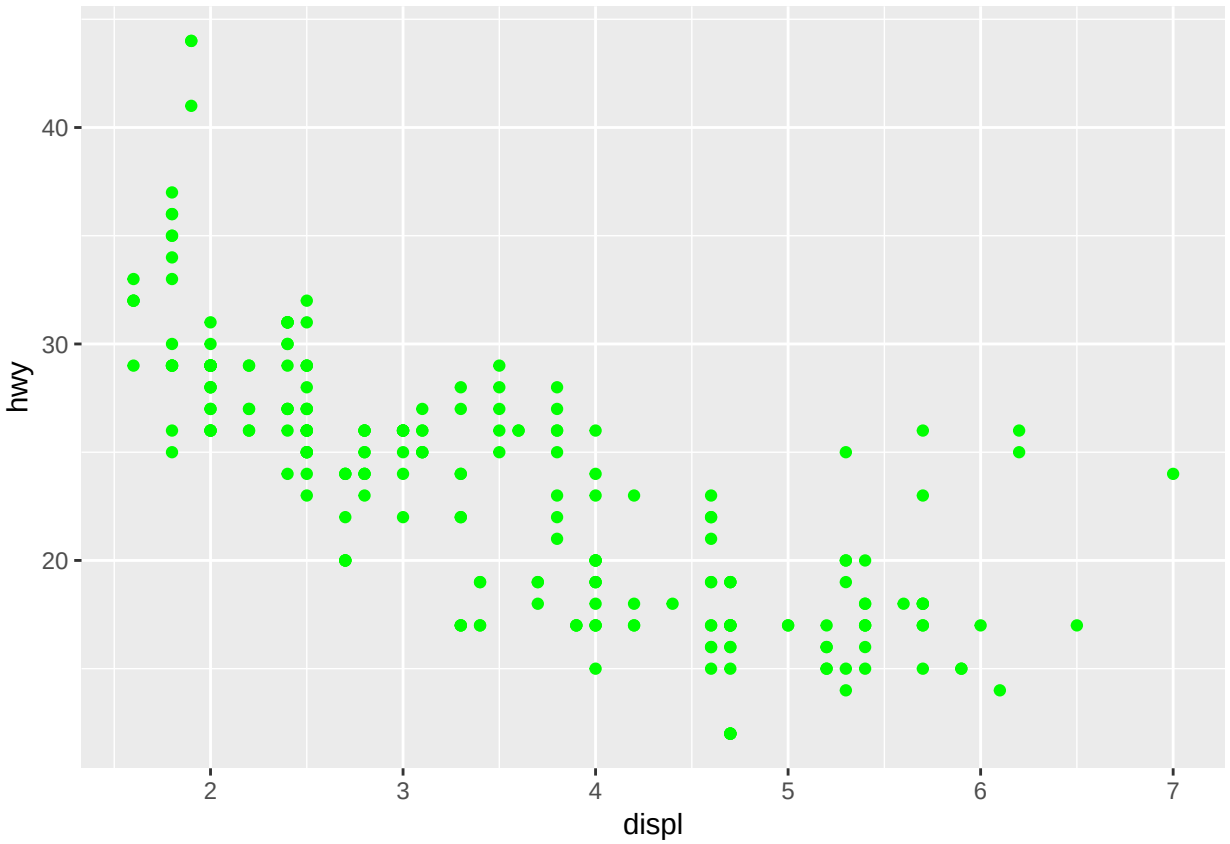
1.2.1.1 Solución

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = "green"))
```



Si se desea que los puntos aparezcan de color verde se tiene que escribir el comando de la manera siguiente.

```
ggplot(data = mpg)+
  geom_point(mapping = aes(x=displ, y = hwy), color="green")
```



1.2.2 Pregunta 2.2.

Toma el dataset de `mpg` anterior y di qué variables son categóricas.

1.2.2.1 Solución

Las variables categóricas son el modelo, la marca, la transmisión, la tracción, el tipo de combustible y el tipo de coche ("manufacturer", "model", "trans", "drv", "fl" y "class")

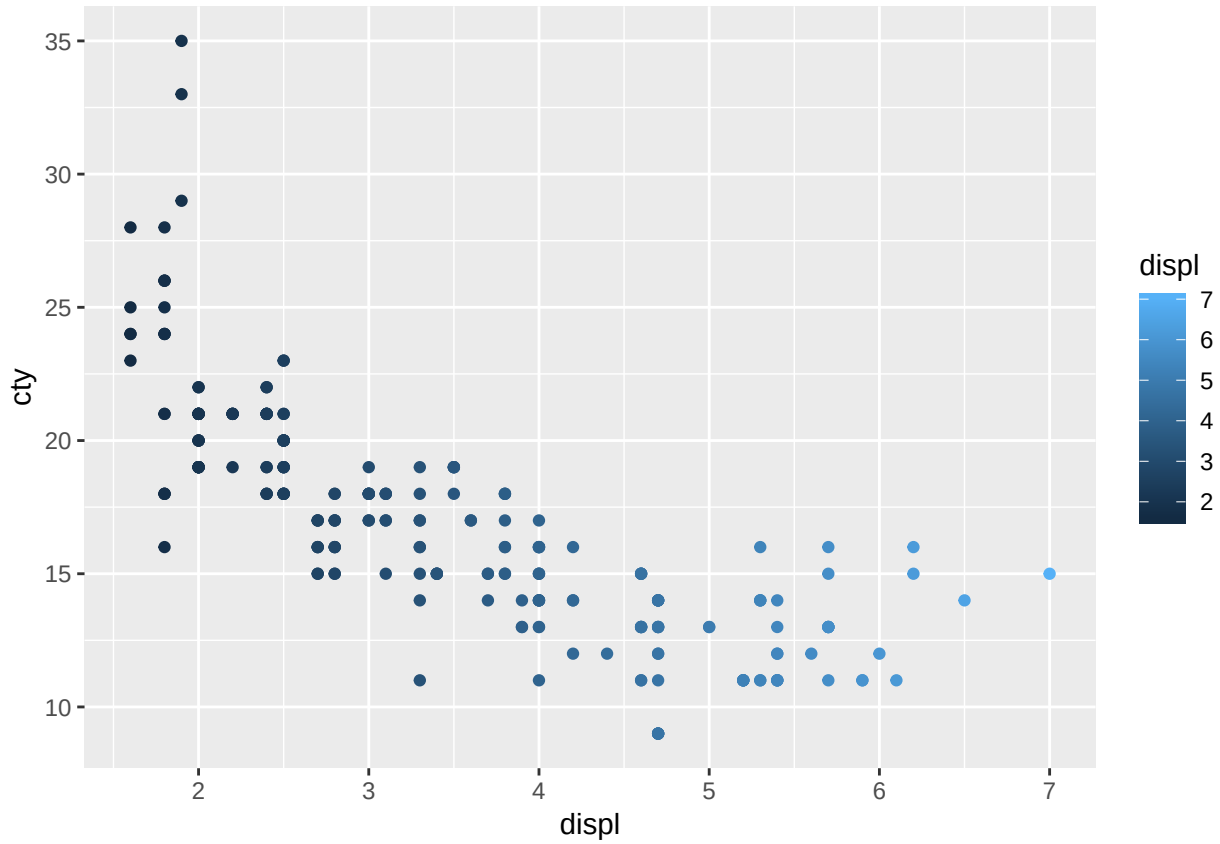
1.2.3 Pregunta 2.3.

Toma el dataset de mpg anterior y di qué variables son continuas. Dibuja las variables continuas con color, tamaño y forma respectivamente.

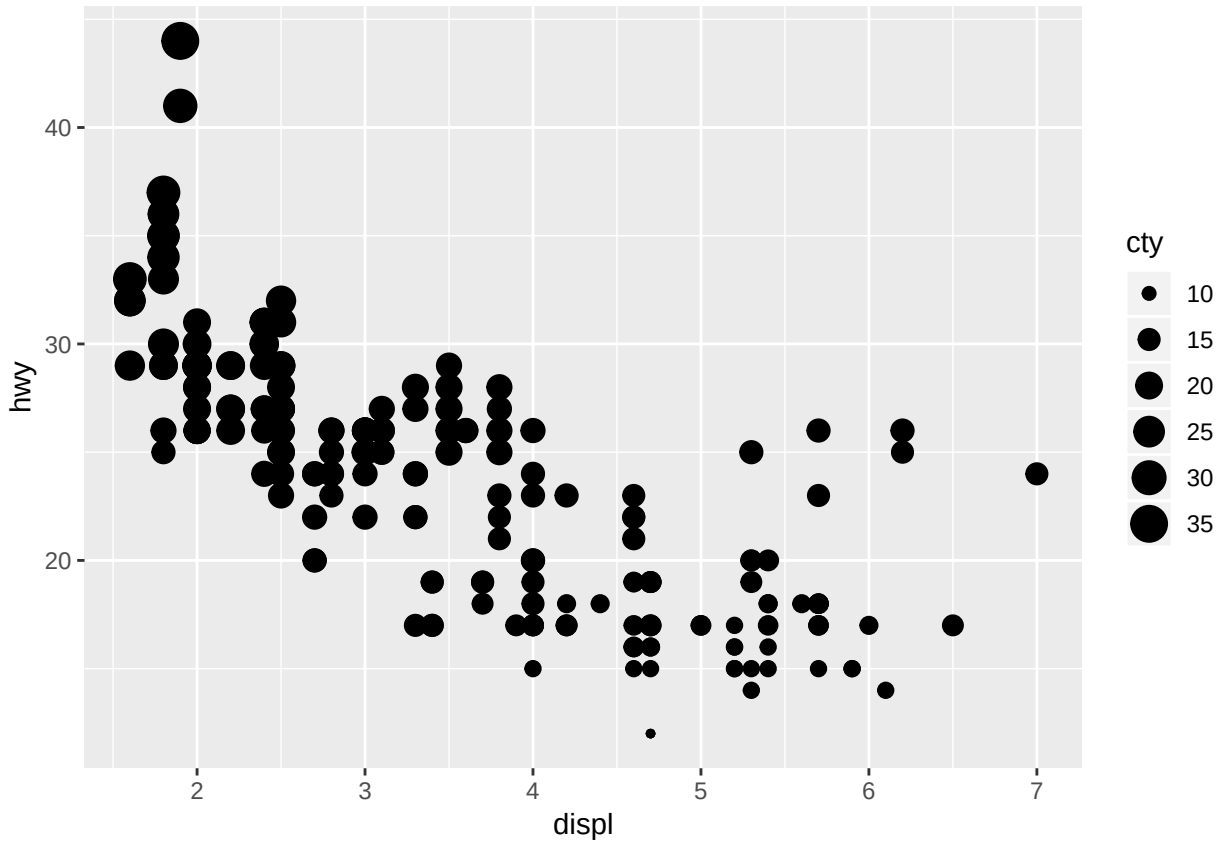
1.2.3.1 Solución

Las variables continuas son "displ","cty" y "hwy"

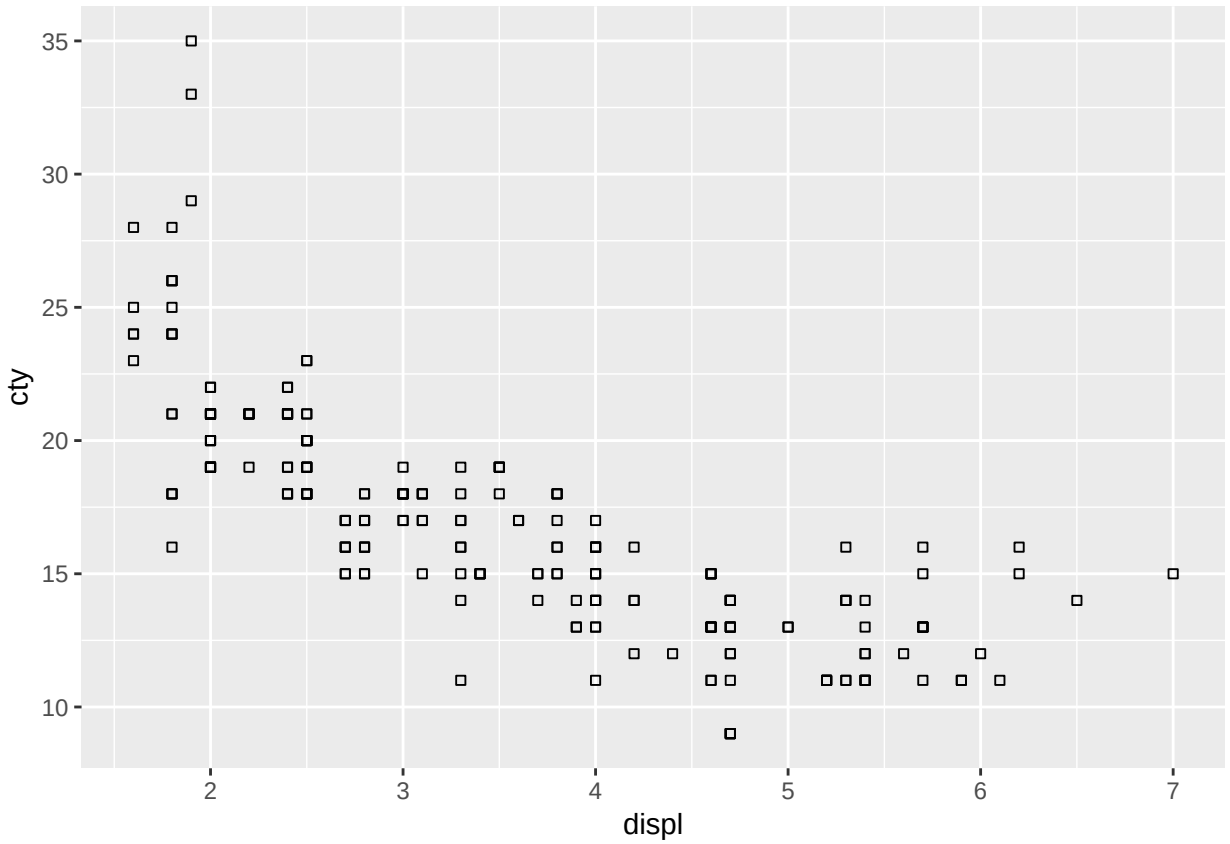
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x=displ , y = cty , color=displ ))
```




```
ggplot(data= mpg) +
  geom_point(mapping = aes(x=displ , y= hwy , size=cty))
```




```
ggplot(data = mpg) +
  geom_point(mapping = aes(x=displ , y =cty ), shape=22)
```



1.2.4 Pregunta 2.4.

¿En qué se diferencian las estéticas para variables continuas y categóricas?

1.2.4.1 Solución

Las variables categóricas indican la pertenencia a un conjunto (la gente rubia, los hijos de María,...) en cambio las variables continuas indican el valor de una cierta propiedad (tiene una capacidad de 20 litros, una altura de 1.8 metros,...).

1.2.5 Pregunta 2.5.

¿Qué ocurre si haces un mapeo de la misma variable a múltiples estéticas?

1.2.5.1 Solución

Todas esas estéticas serán usadas conjuntamente.

1.2.6 Pregunta 2.6.

Vamos a conocer una estética nueva llamada `stroke`. ¿Qué hace? ¿Con Qué formas funciona bien?

1.2.6.1 Solución

La estética 'stroke' permite modificar el grosor del borde de aquellas figuras que lo tengan ('shape' entre 21 y 25).

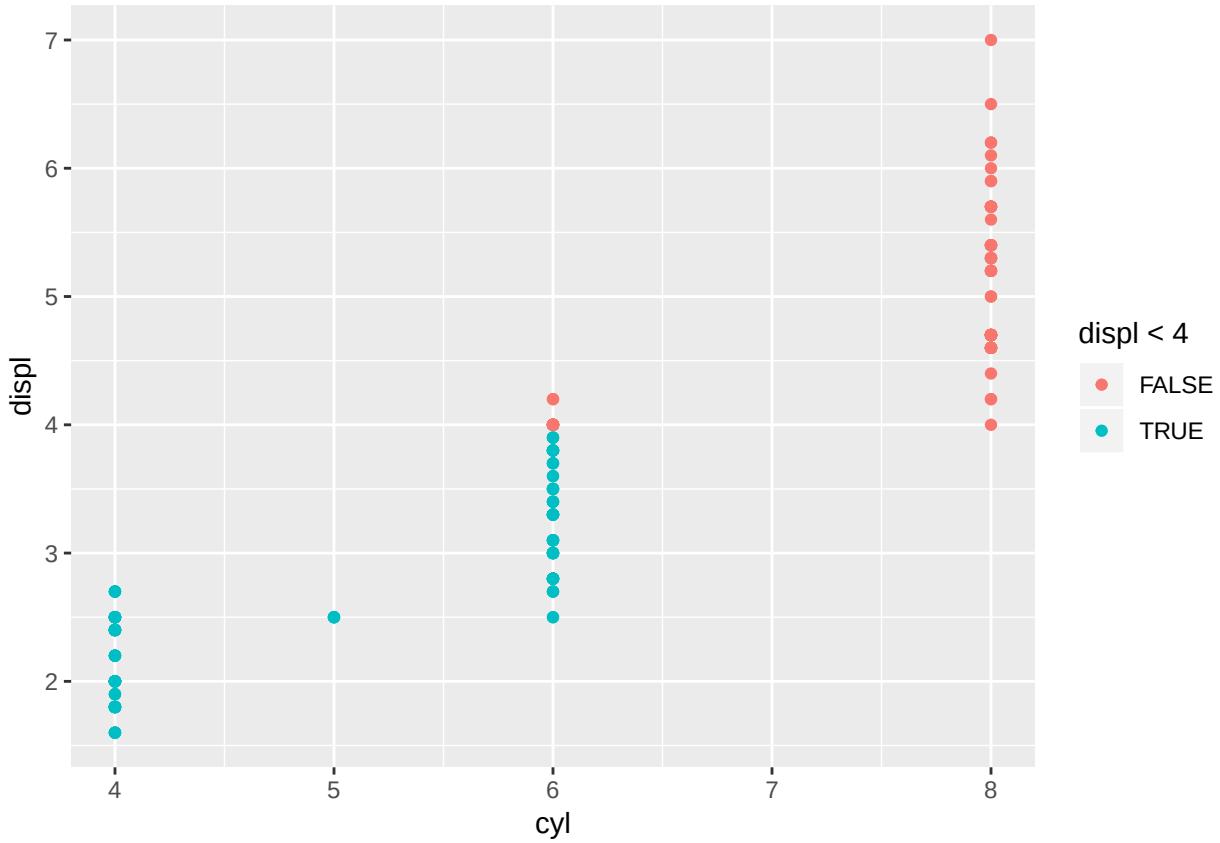
1.2.7 Pregunta 2.7.

¿Qué ocurre si haces un mapeo de una estética a algo que no sea directamente el nombre de una variable (por ejemplo aes(color = displ < 4))?

1.2.7.1 Solución

R entiende el criterio como una división y aplica la estética a aquello que comple la condición impuesta.

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x= cyl , y = displ ,color = displ < 4))
```



1.3 Tarea 3: Subplots con facets. Sección 3: Lecciones 18 y 19

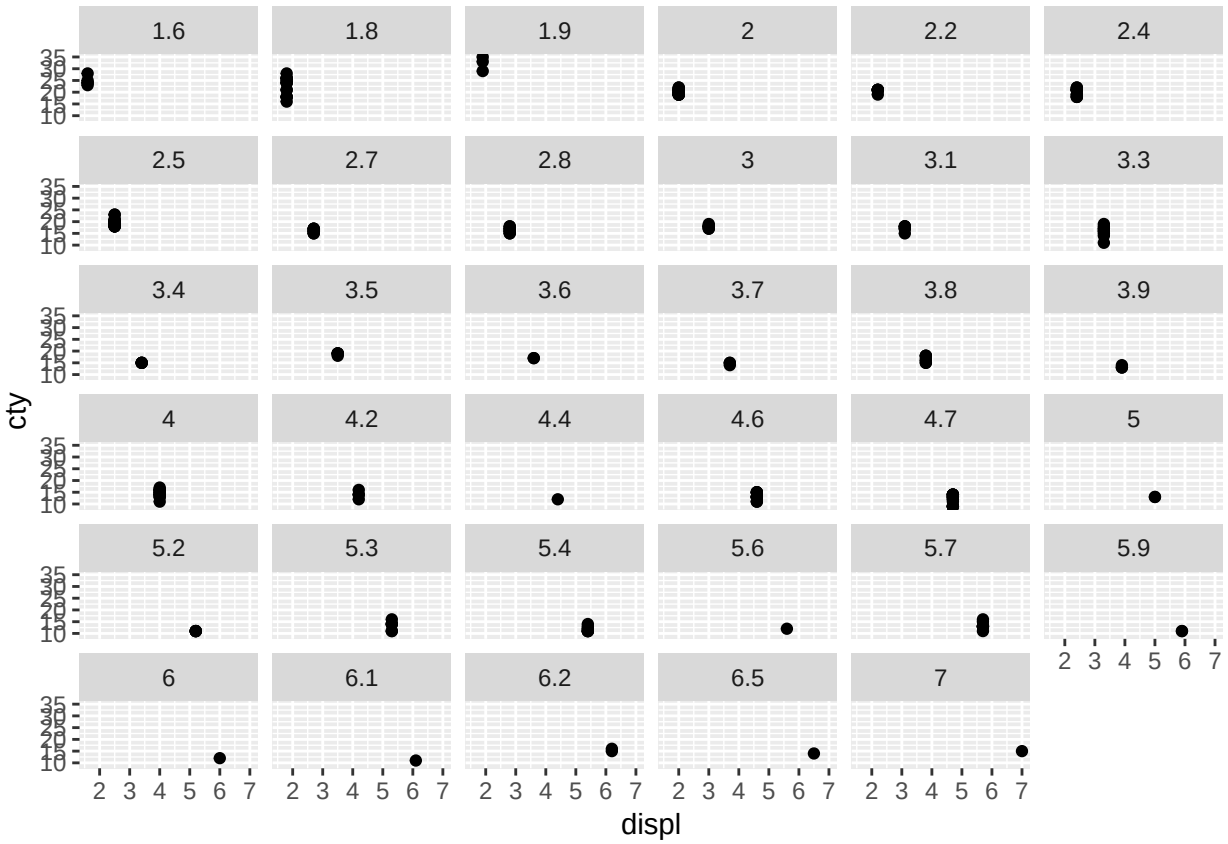
1.3.1 Pregunta 3.1.

¿Qué ocurre si hacemos un `facet` de una variable continua?

1.3.1.1 Solución

Hará tantos cuadros como valores encuentre.

```
ggplot(data = mpg) +
  geom_point( mapping = aes(x=displ , y = cty )) + facet_wrap(~displ)
```



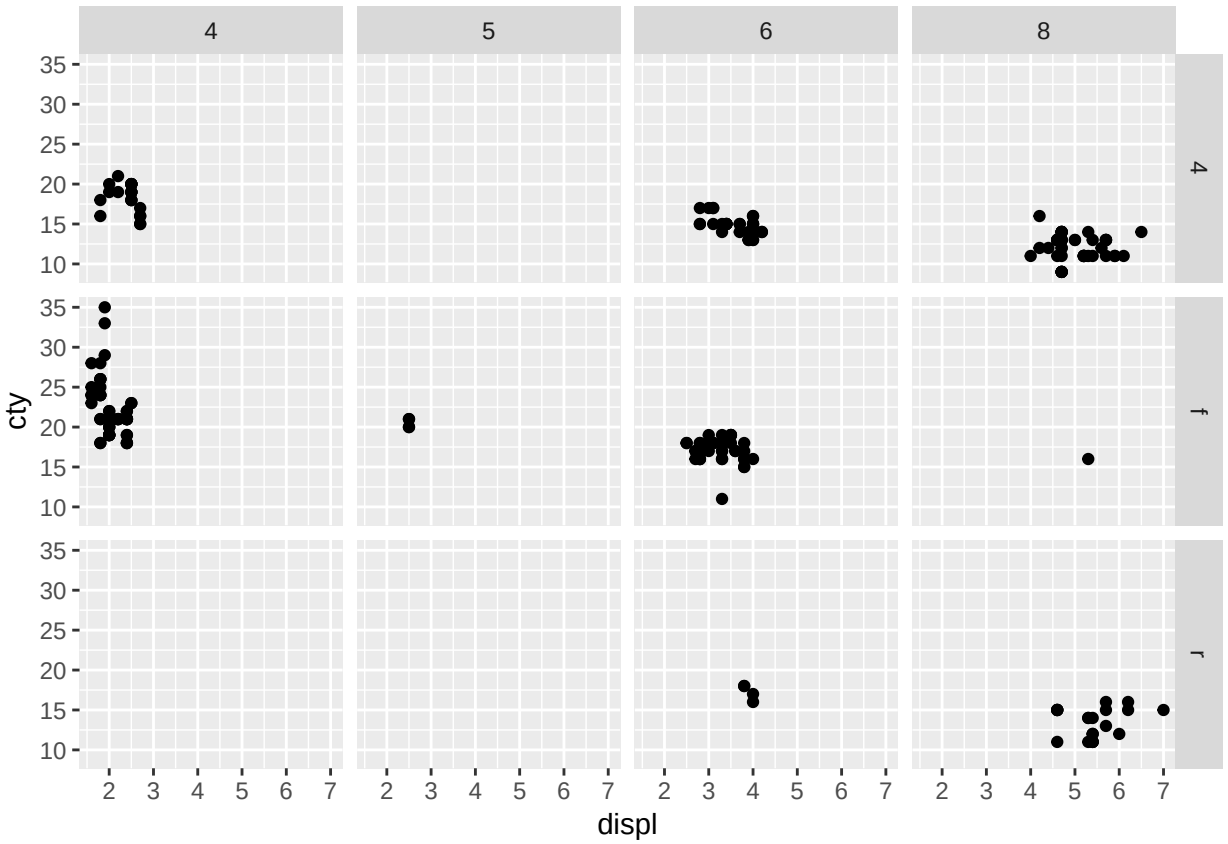
1.3.2 Pregunta 3.2.

¿Qué significa si alguna celda queda vacía en el gráfico `facet_grid(drv~cyl)`? ¿Qué relación guardan esos huecos vacíos con el gráfico siguiente?

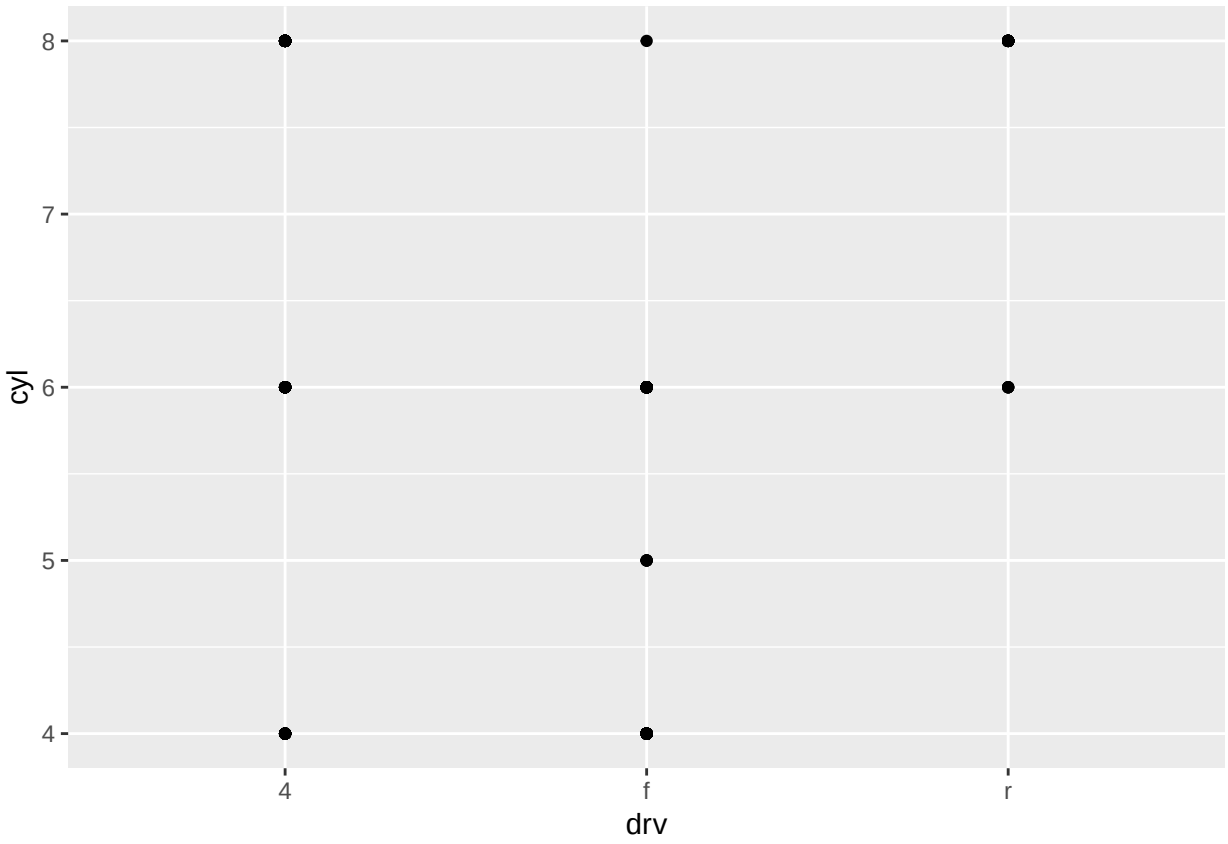
1.3.2.1 Solución

Hay huecos vacíos debido a que no hay datos que satisfagan ambas condiciones. La tabla posterior muestra esos vacíos.

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x= displ , y = cty))+facet_grid(drv~cyl)
```




```
ggplot(data = mpg) +
  geom_point(mapping = aes(x=drv, y = cyl))
```



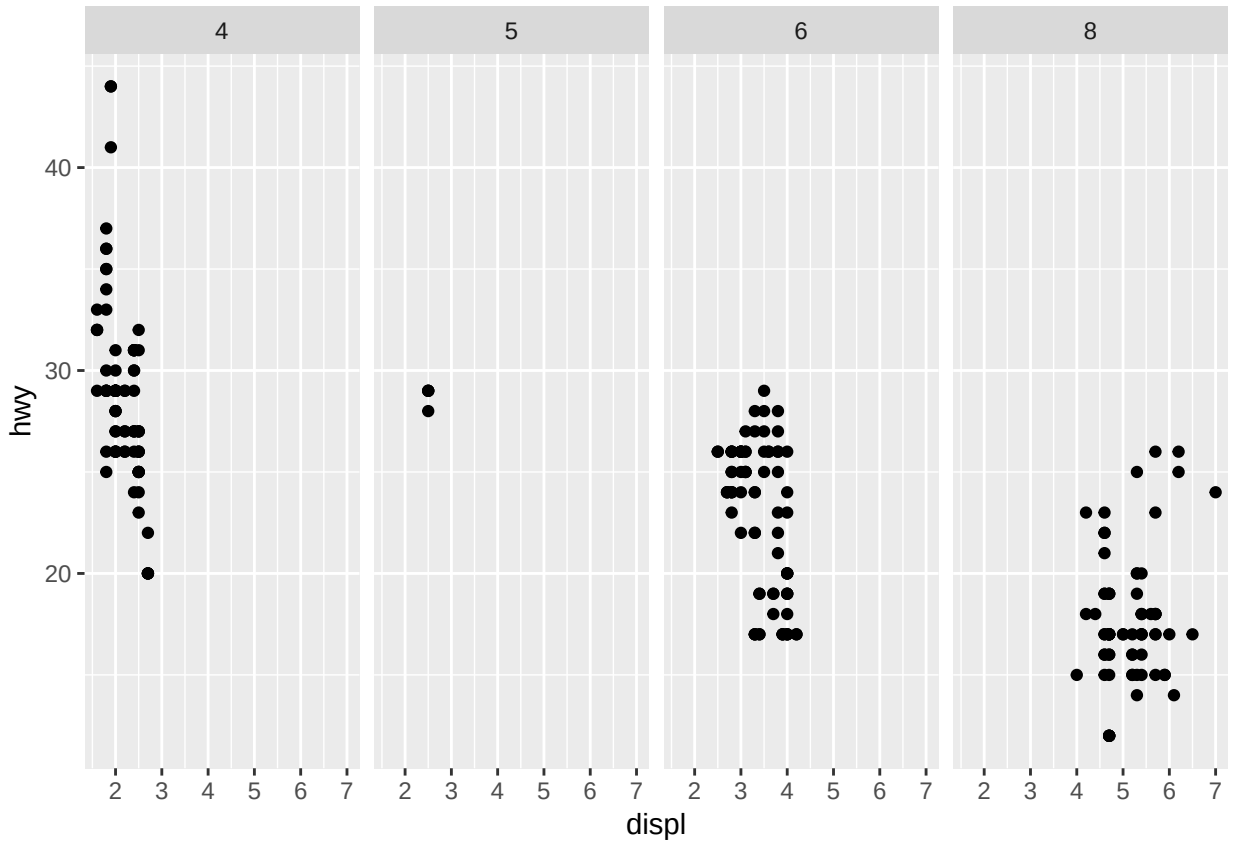
1.3.3 Pregunta 3.3.

¿Qué gráficos generan las siguientes dos instrucciones? ¿Qué hace el punto? ¿Qué diferencias hay de escribir la variable antes o después de la vírgulilla ("~")?

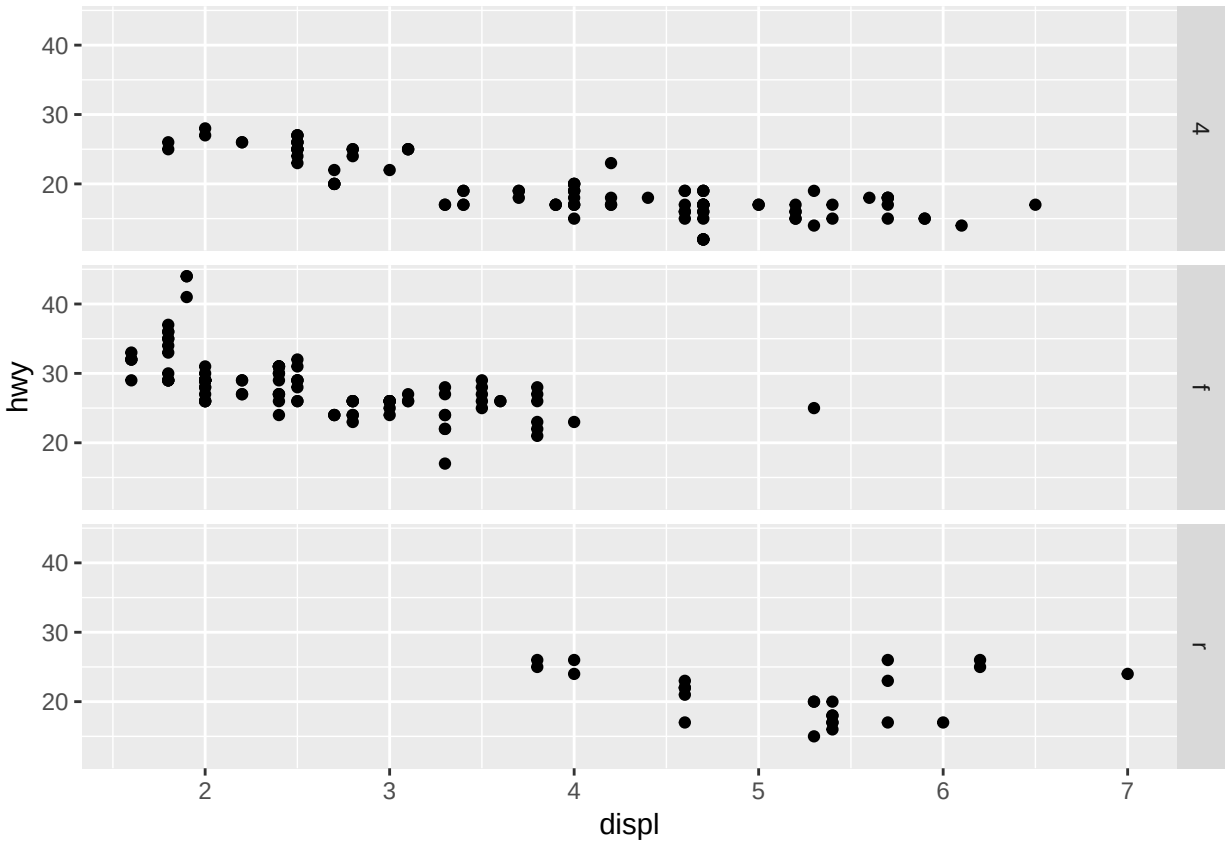
1.3.3.1 Solución

Representa los puntos clasificados por los datos "cyl" y "drv" respectivamente. Escribir antes o después de la vírgula marca si la división se visualizara por filas o por columnas.

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x=displ, y = hwy)) +
  facet_grid(.~cyl)
```



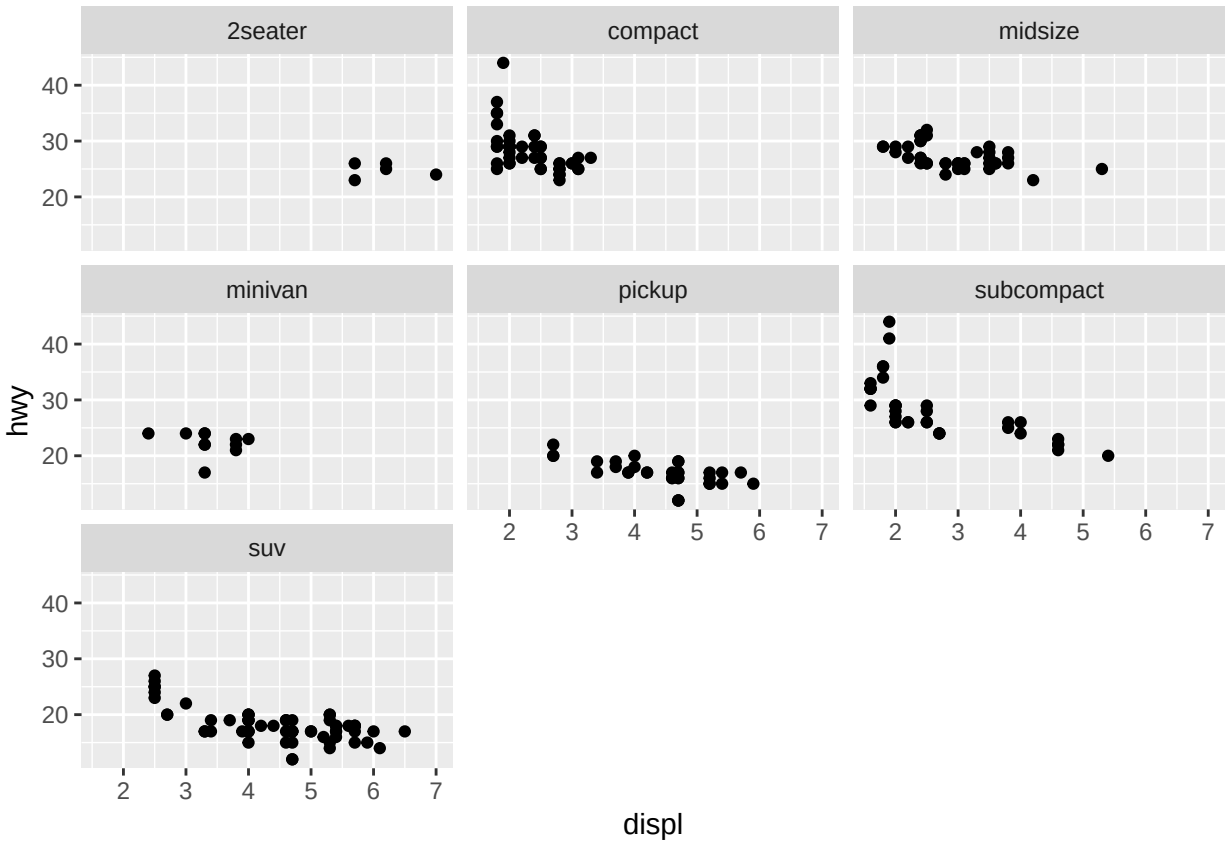

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x=displ, y = hwy)) +
  facet_grid(drv~.)
```



1.3.4 Pregunta 3.4.

**El primer facet que hemos pintado era el siguiente:

```
ggplot(data = mpg) +
 geom_point(mapping = aes(x = displ, y = hwy)) +
 facet_wrap(~class, nrow = 3)
```



¿Qué ventajas crees que tiene usar facets en lugar de la estética del color? ¿Qué desventajas? ¿Qué cambiará si tu dataset fuera mucho más grande?**

1.3.4.1 Solución

La estética de color está limitada por la cantidad de colores disponibles diferenciables, en cambio independiente de la cantidad de datos si estos son divididos en diferentes gráficas estos serán fáciles de visualizar.

1.4 Tarea 4: Geometrías con ggplot2. Sección 3: Lecciones 20 y 21.

Repasa los contenidos de las geometrías de ggplot2 y mira a ver si sabes responder a las siguientes preguntas. Preguntas de esta tarea

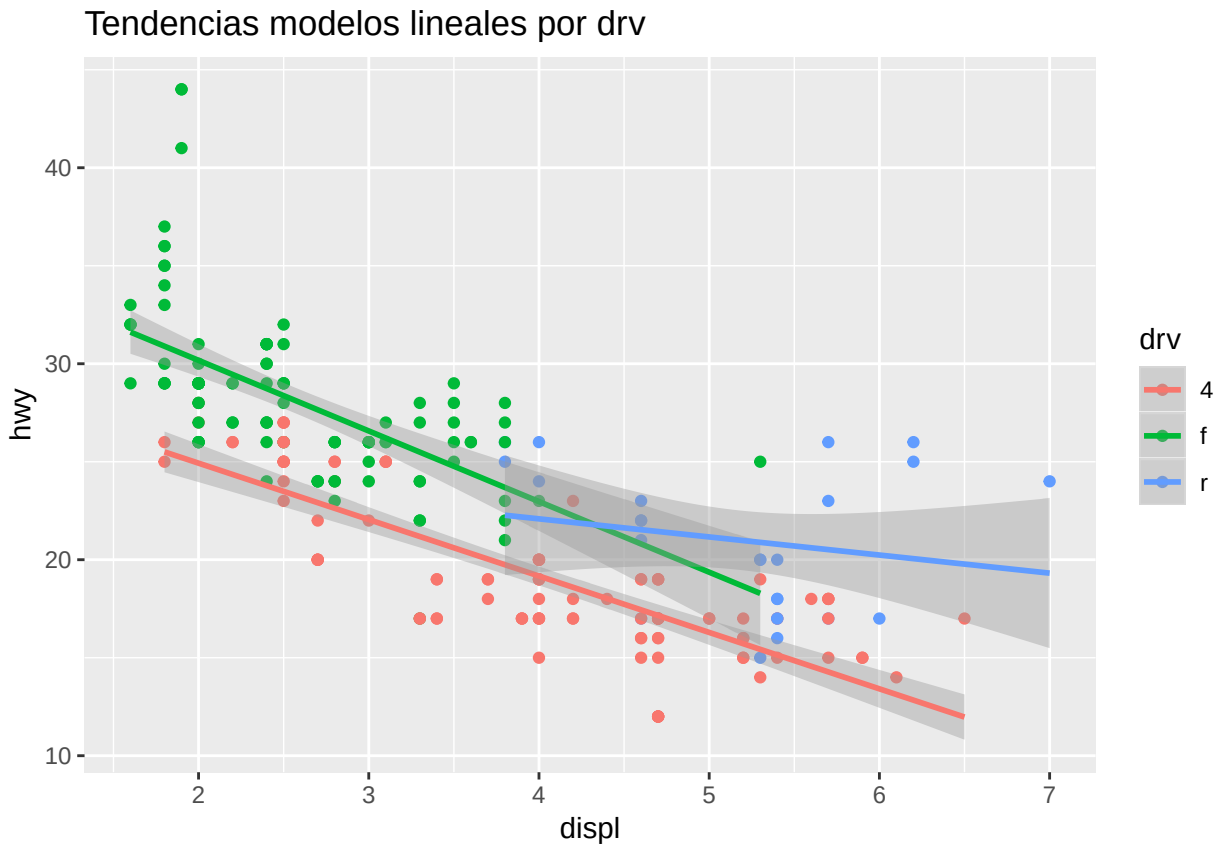
1.4.1 Cuestión 4.1.

Ejecuta este código mentalmente y predice el resultado. Luego ejecutalo en R y comprueba tu hipótesis:

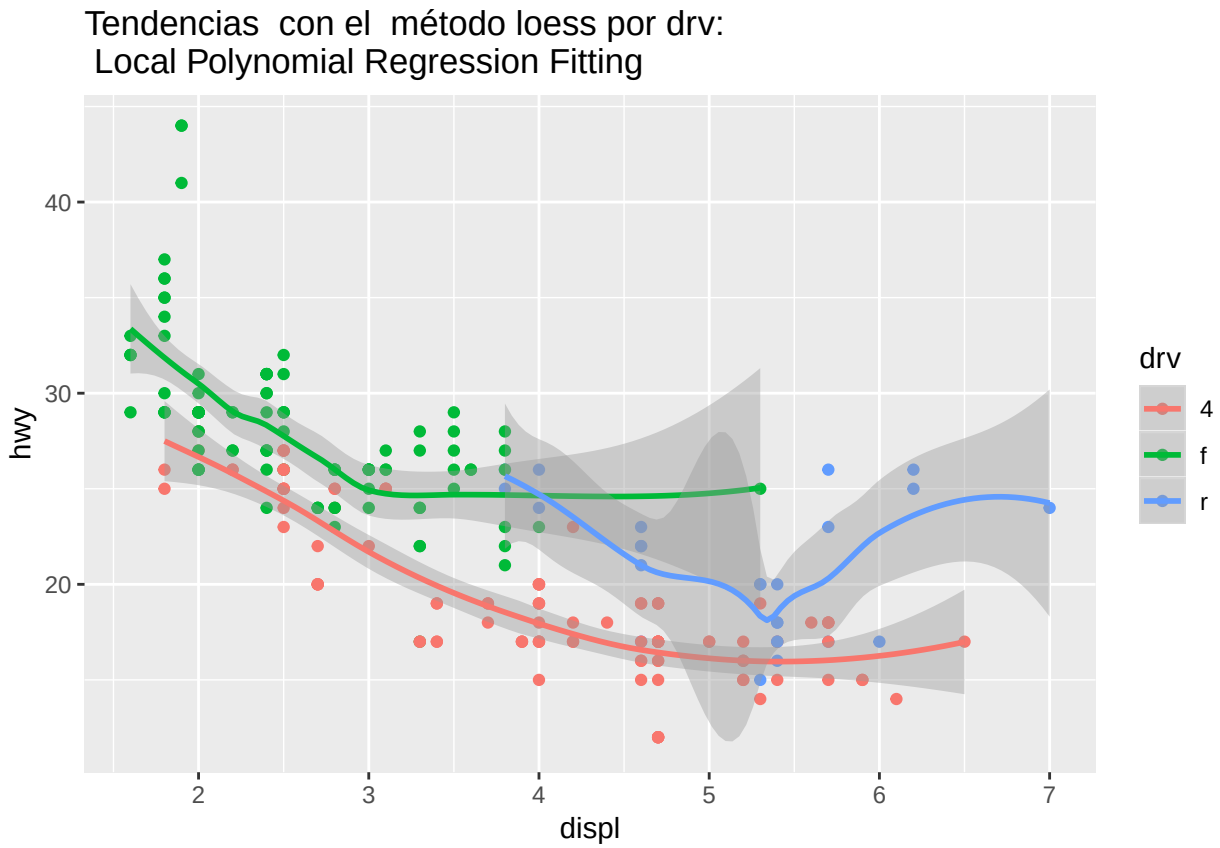
```
ggplot(data = mpg, mapping = aes(x=displ, y = hwy,color = drv)) +
  geom_point() +
  geom_smooth( se = F)
```

1.4.1.1 Solución

```
gg1=ggplot(data = mpg, mapping = aes(x=displ, y = hwy,color = drv)) +geom_point()
gg1+geom_smooth(method="lm",se=TRUE)+labs(title="Tendencias modelos lineales por drv")
```

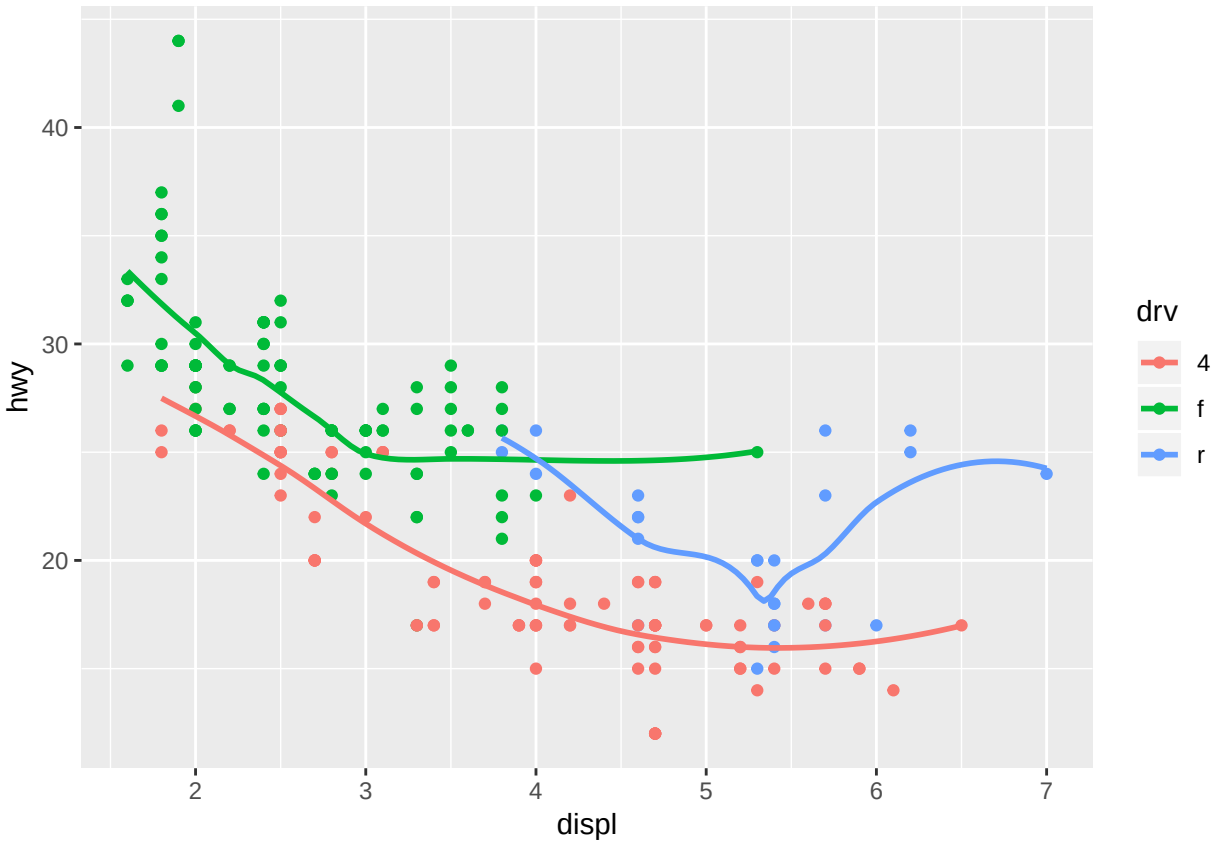



```
gg1+geom_smooth(method="loess",se = TRUE)+labs(title="Tendencias  con el  método loess por drv:\n Local
```




```
ggplot(data = mpg, mapping = aes(x=displ, y = hwy, color = drv)) +
  geom_point() +
  geom_smooth( se = F)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



1.4.2 Cuestión 4.2.

¿Qué hace el parámetro show.legend = F? ¿Qué pasa si lo eliminamos? ¿Cuando lo añadirías y cuando lo quitarías?

1.4.2.1 Solución

No sale la leyenda, si lo eliminamos (si existe) sale la leyenda. La leyenda es necesaria cuando hay decoraciones en el dibujo que vengan asociada a alguna otra variable. En cualquier caso hay que poner la leyenda si es necesaria para la iterpretación del gráfico.

1.4.3 Cuestión 4.3.

¿Qué hace el parámetro se de la función geom_smooth()? ¿Qué pasa si lo eliminamos? ¿Cuando lo añadirías y cuando lo quitarías?

1.4.3.1 Solución

Es el parámetro `se`. Es un parámetro lógico que muestra bandas de confianza asociada para a la variable estimada. Depende del method utilizado para el suavizado.

1.4.4 Cuestión 4.4.

Describe qué hacen los dos siguientes gráficos y di si serán igual y diferente. Justifica tu respuesta.

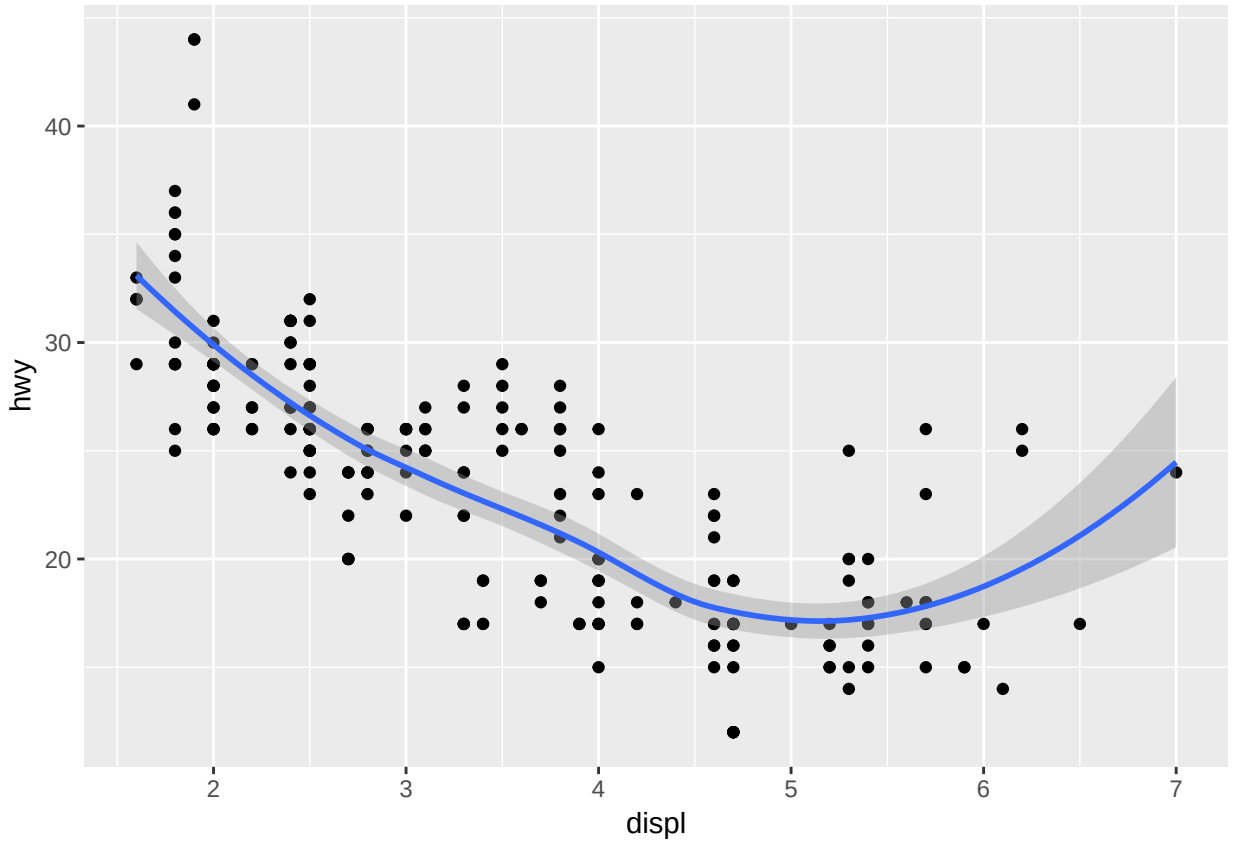
```
ggplot(data = mpg, mapping = aes(x=displ, y = hwy)) +
  geom_point() +
  geom_smooth()
```

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x=displ, y = hwy)) +
  geom_smooth(mapping = aes(x=displ, y = hwy))
```

1.4.4.1 Solución

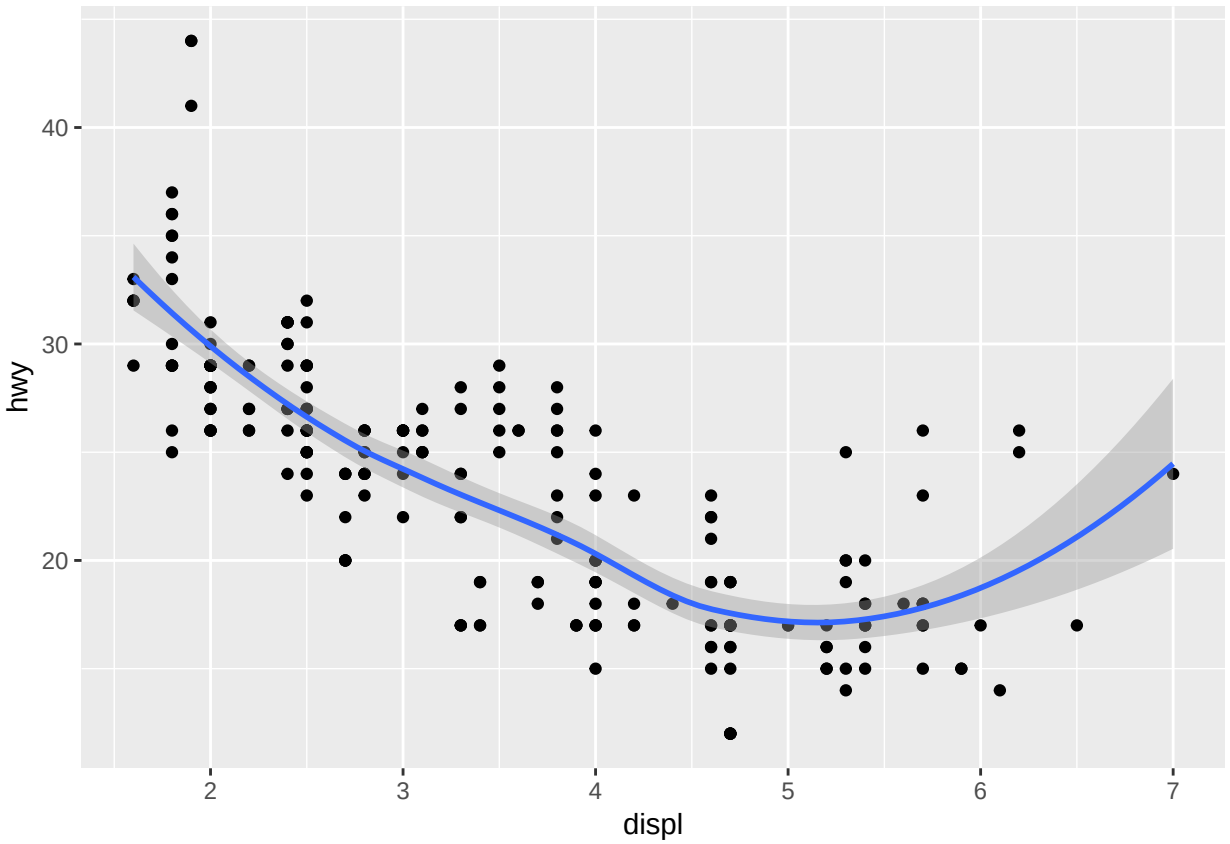
```
ggplot(data = mpg, mapping = aes(x=displ, y = hwy)) +
  geom_point() +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



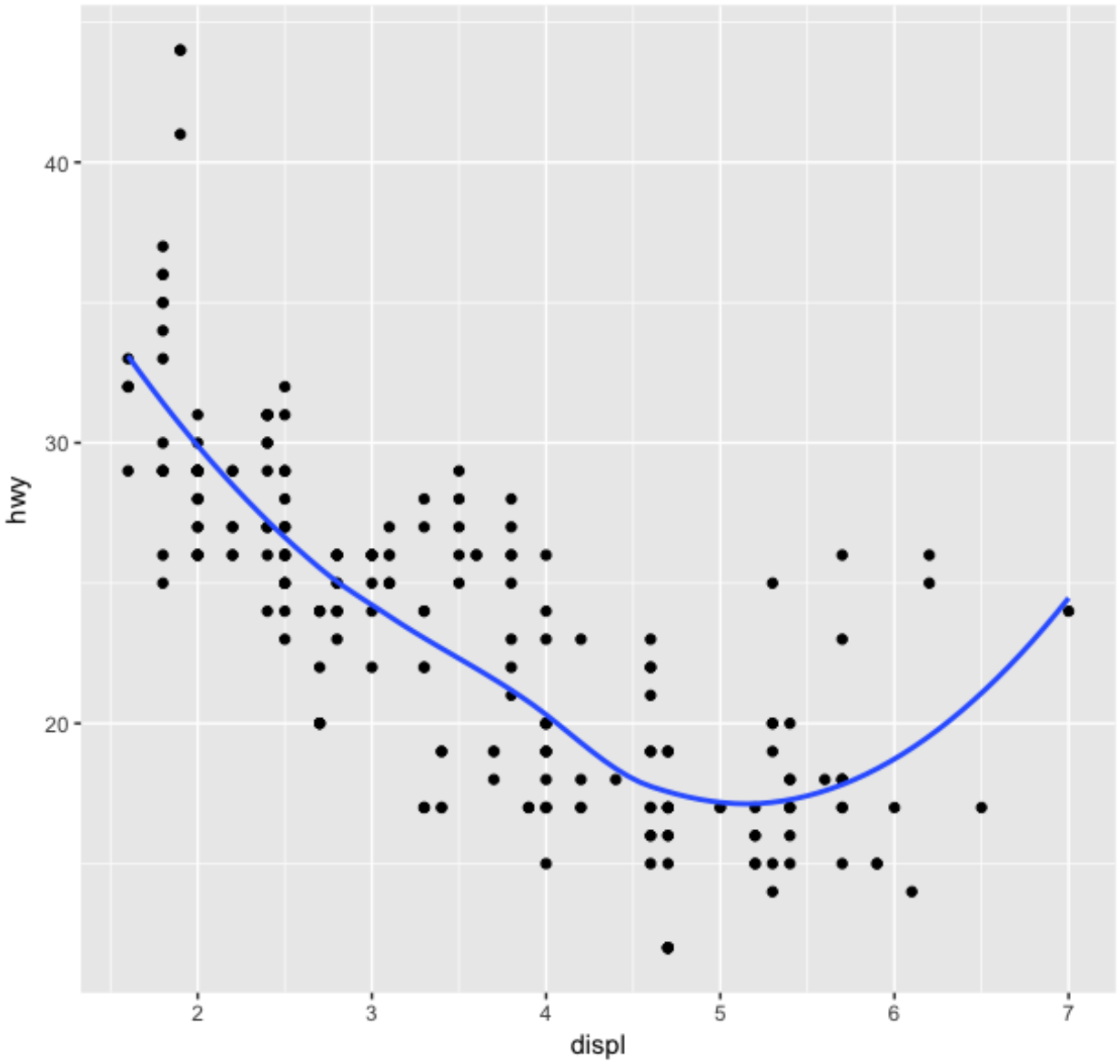

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x=displ, y = hwy)) +
  geom_smooth(mapping = aes(x=displ, y = hwy))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



1.4.5 Cuestión 4.5.

Reproduce el código de R que te genera el siguiente gráfico.

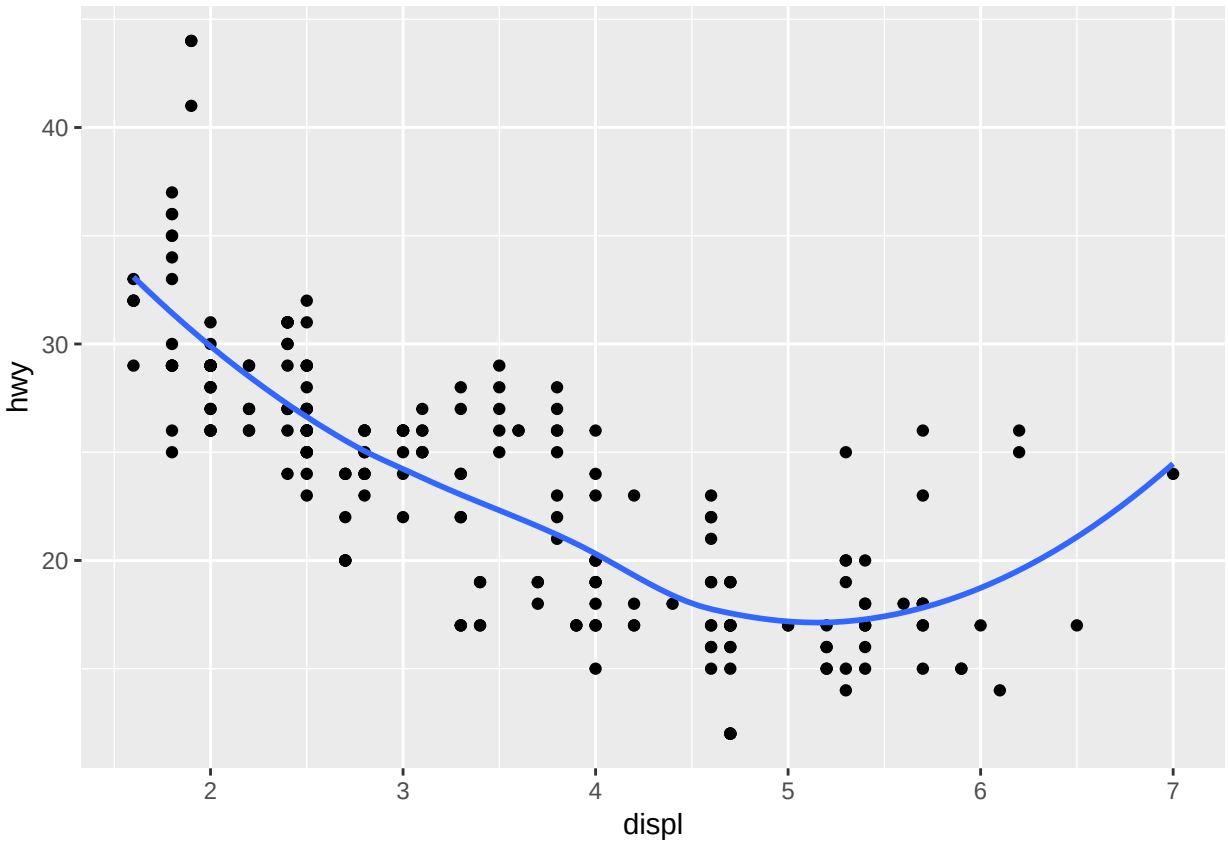


1.4.5.1 Solución

Por ejemplo:

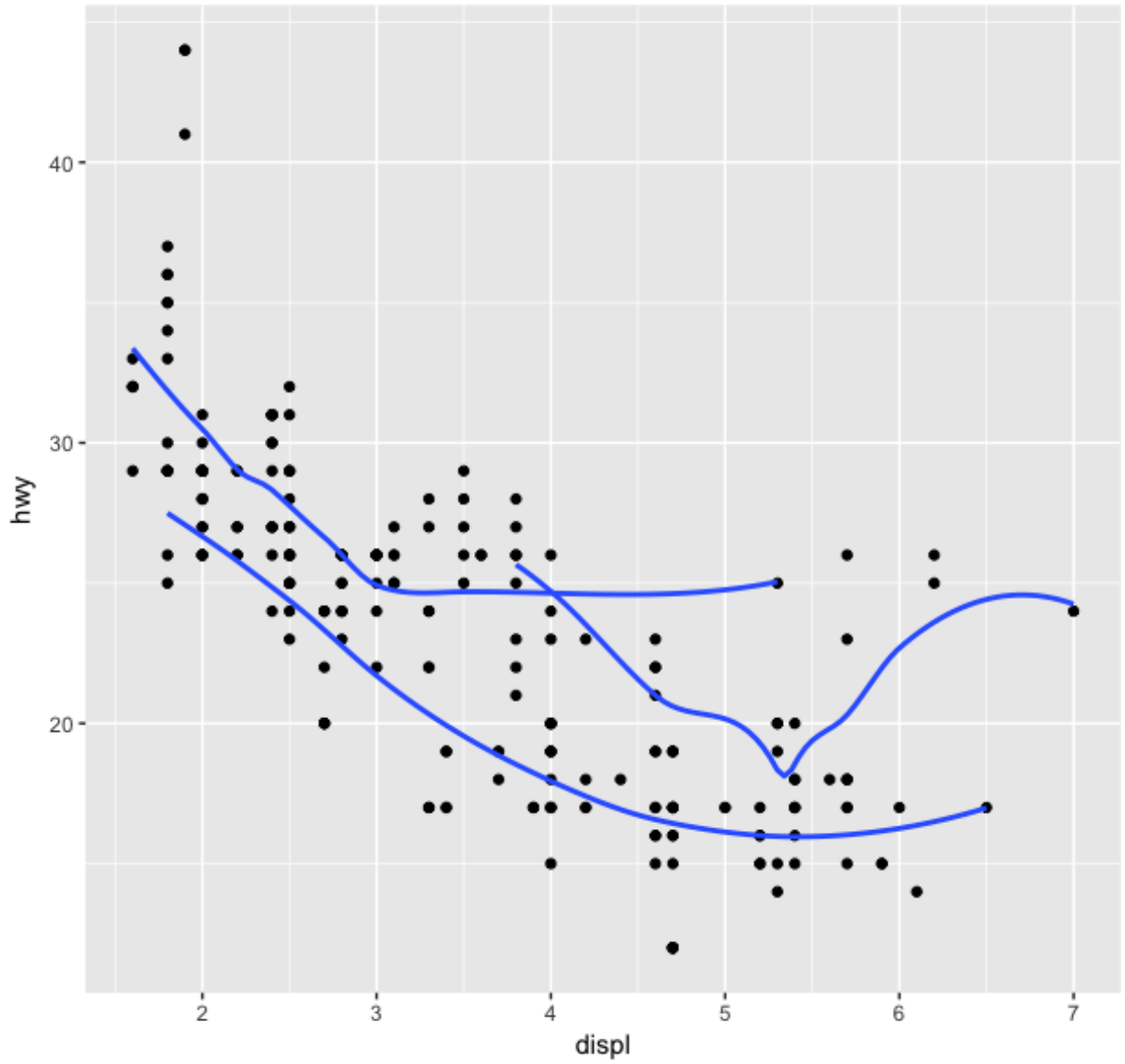
```
ggplot(data = mpg, mapping = aes(x=displ, y = hwy)) +
  geom_point() +
  geom_smooth(se=FALSE)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



1.4.6 Cuestión 4.6.

Reproduce el código de R que te genera el siguiente gráfico.

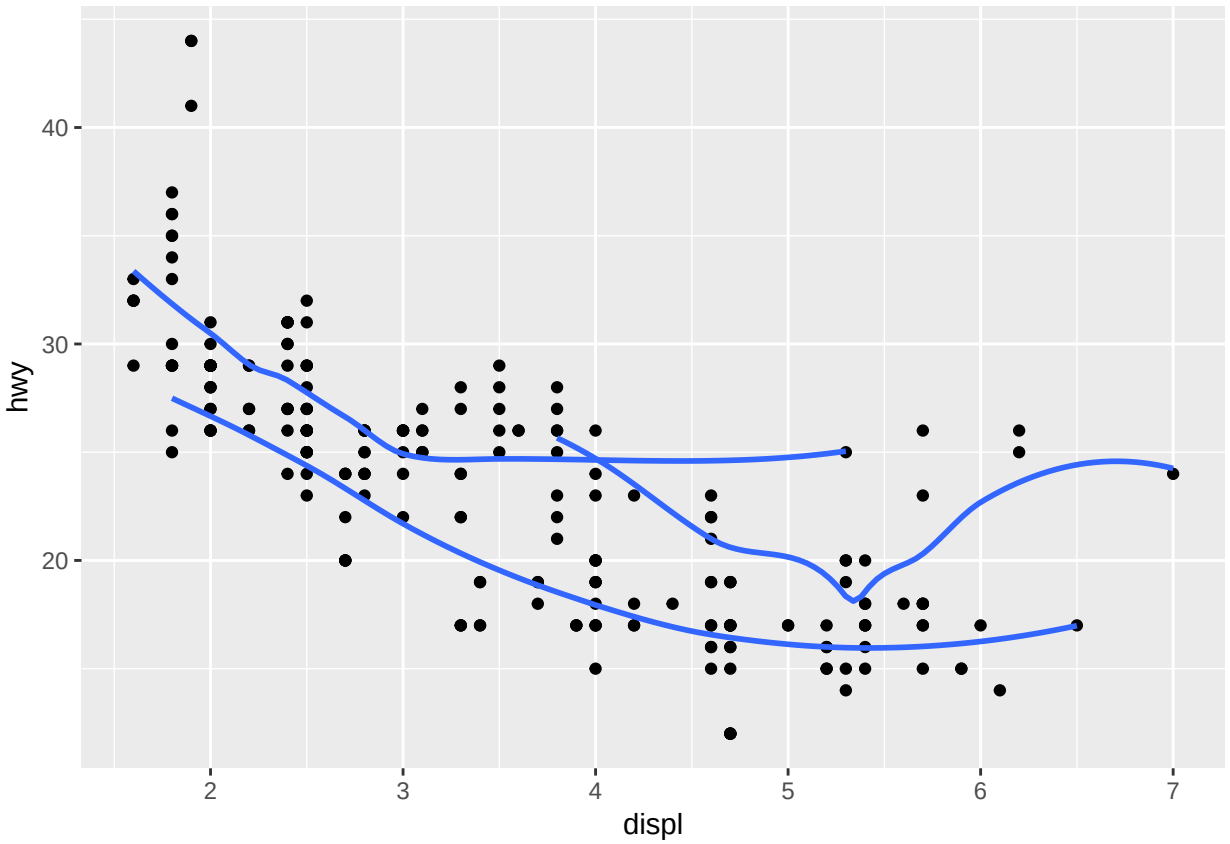


1.4.6.1 Solución

Por ejemplo:

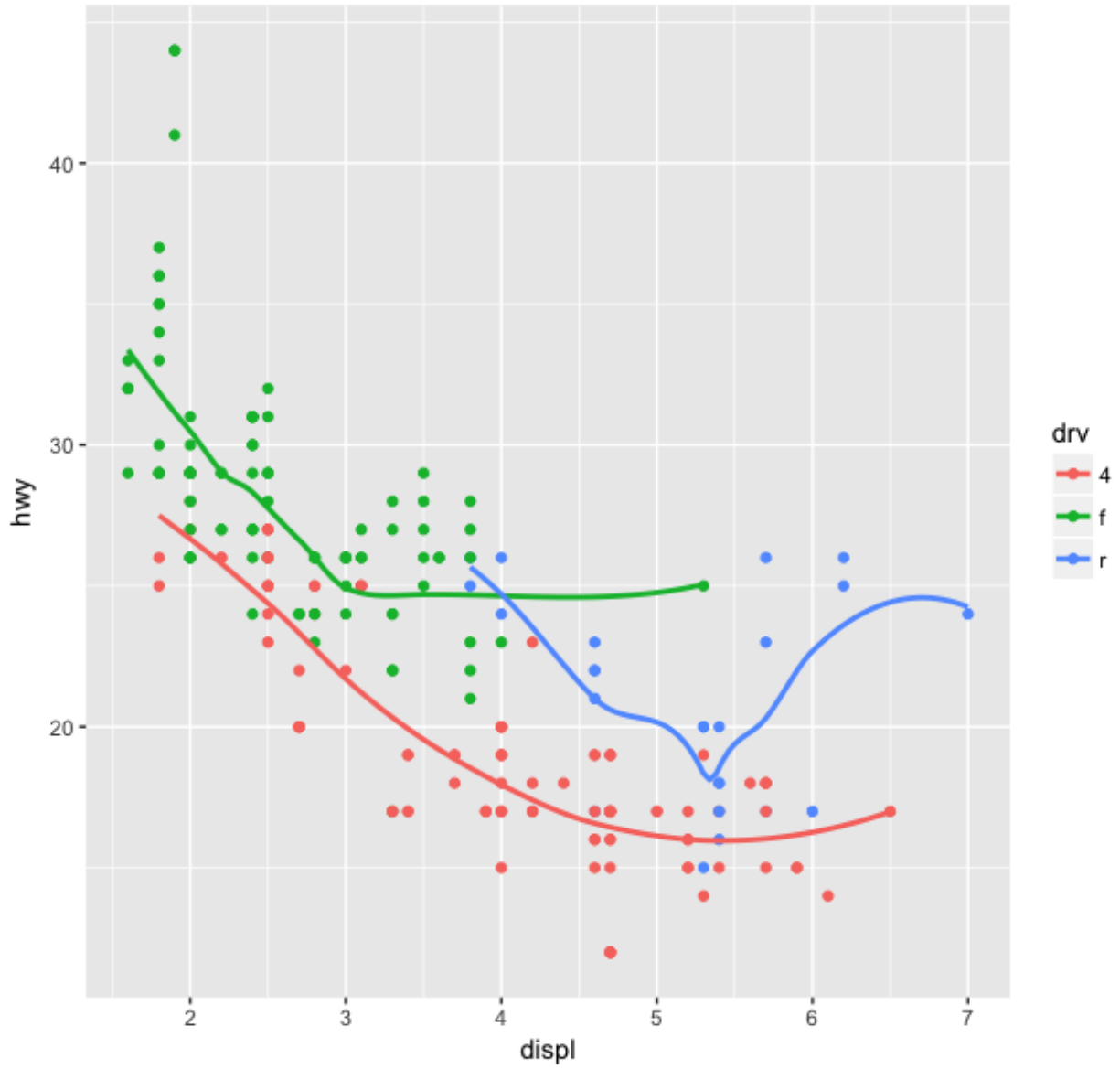
```
ggplot(data = mpg, mapping = aes(x=displ, y = hwy,group=drv)) +
  geom_point() +
  geom_smooth(se=FALSE)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



1.4.7 Cuestión 4.7.

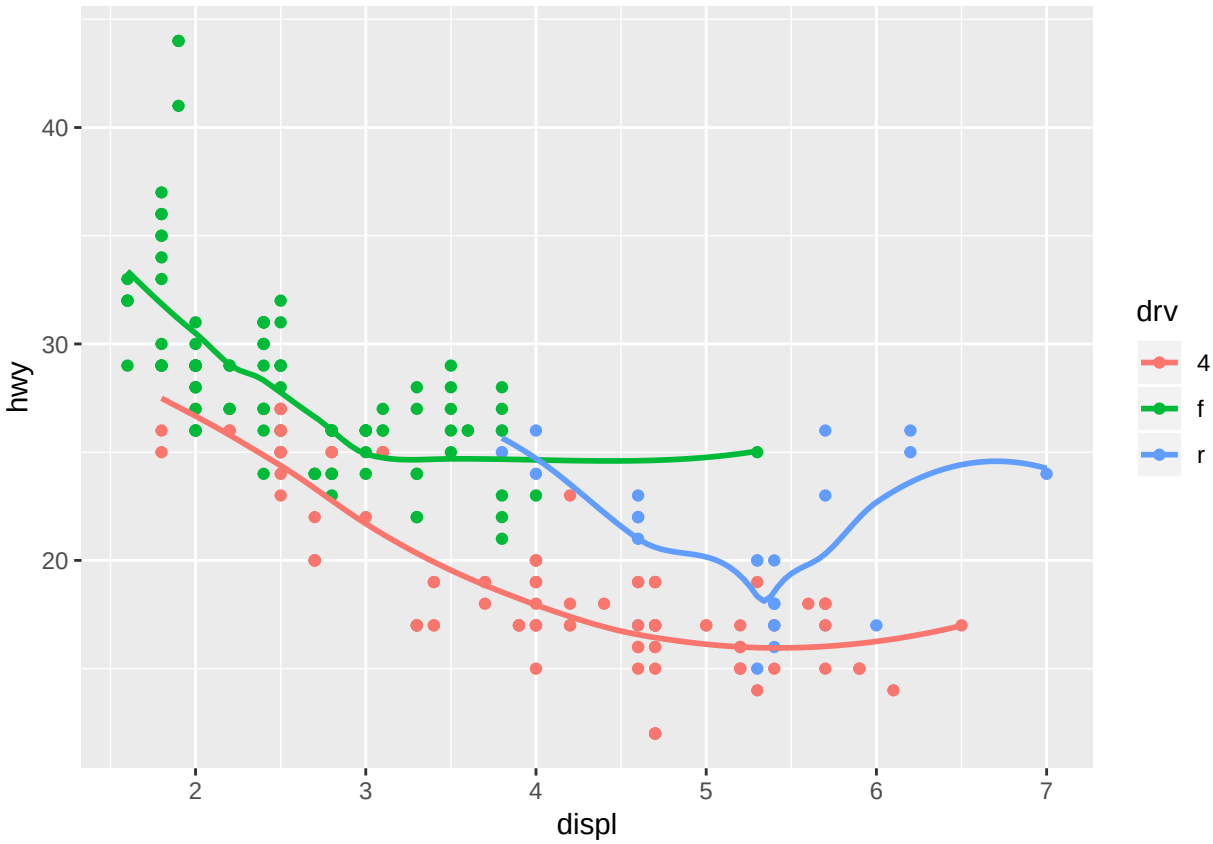
Reproduce el código de R que te genera el siguiente gráfico.



1.4.7.1 Solución

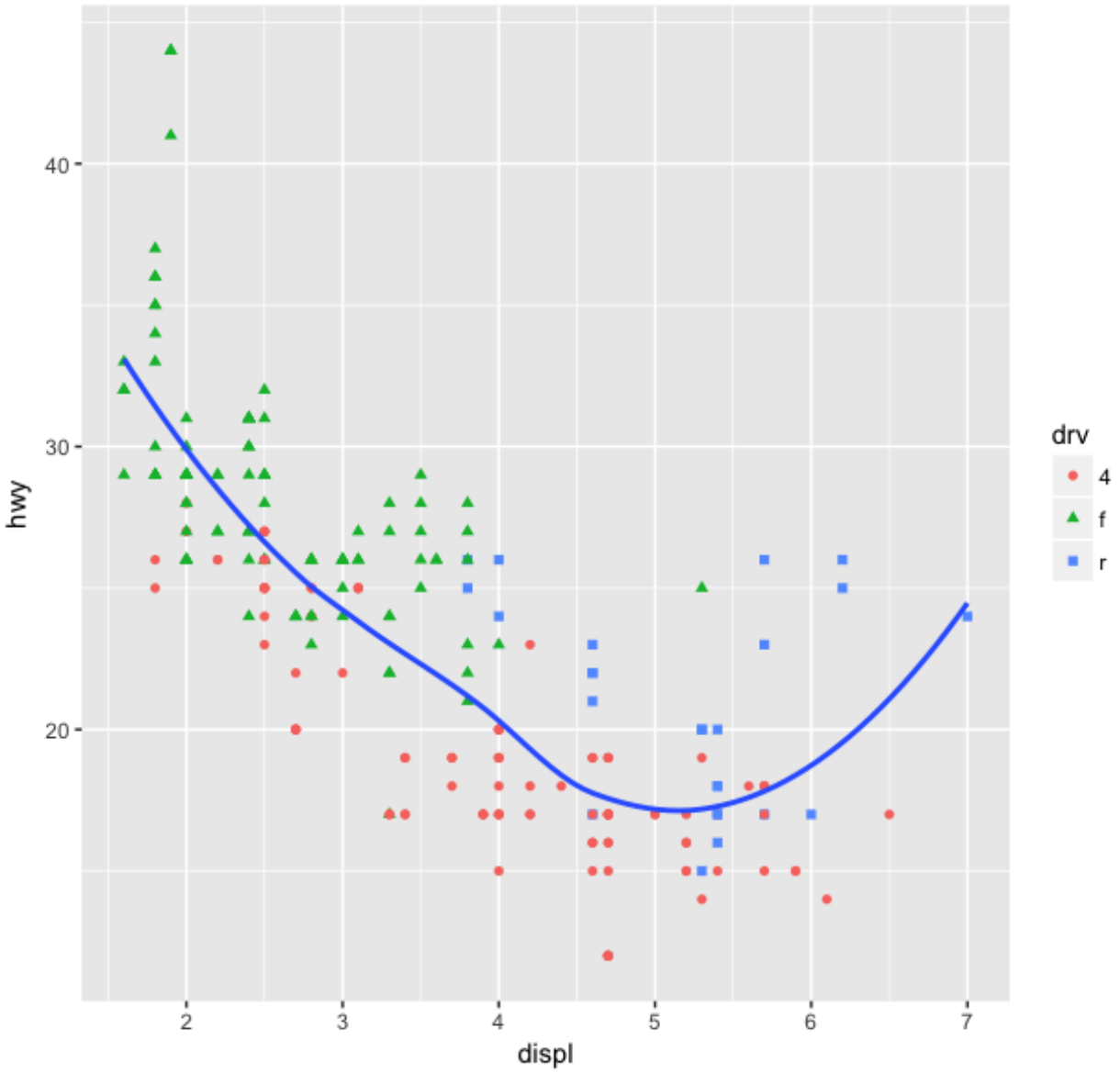
```
ggplot(data = mpg, mapping = aes(x=displ, y = hwy,color=drv)) +
  geom_point() +
  geom_smooth(se=FALSE)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



1.4.8 Cuestión 4.8.

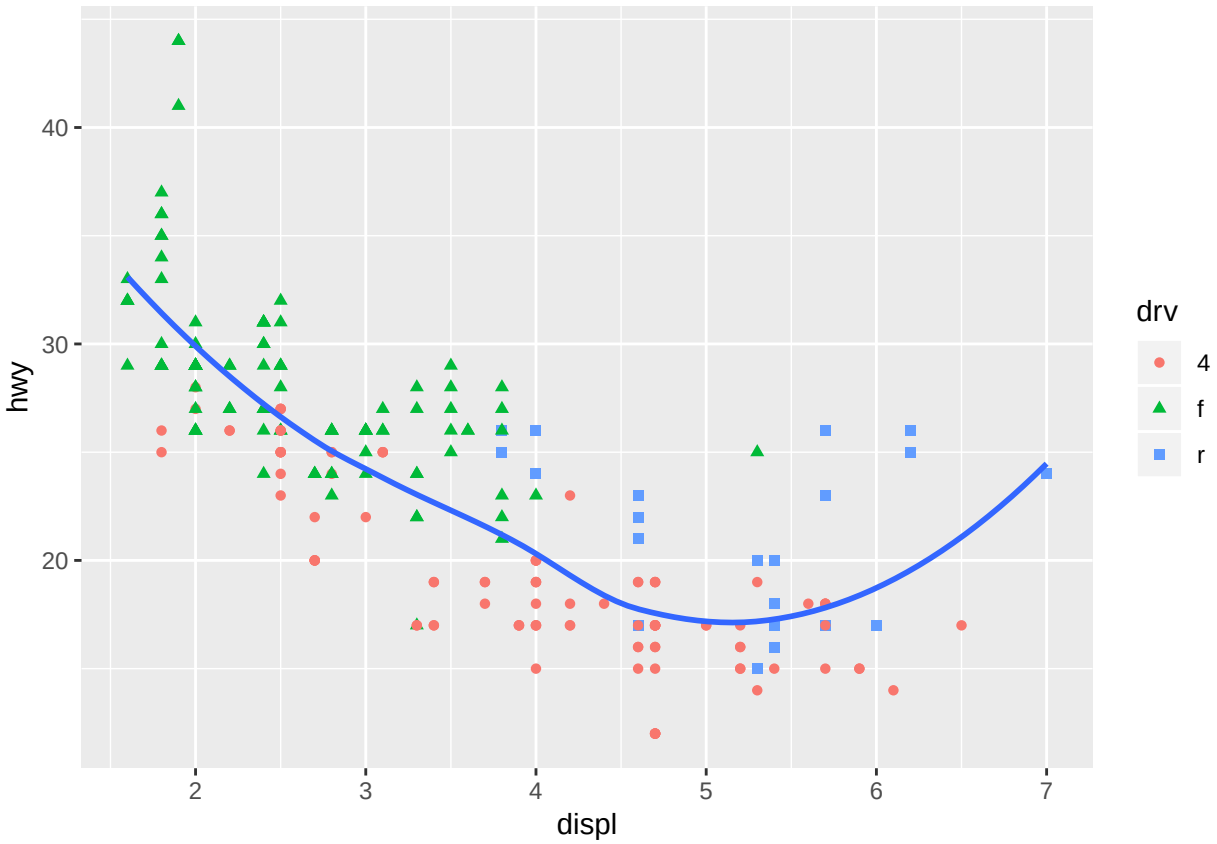
Reproduce el código de R que te genera el siguiente gráfico.



1.4.8.1 Solución

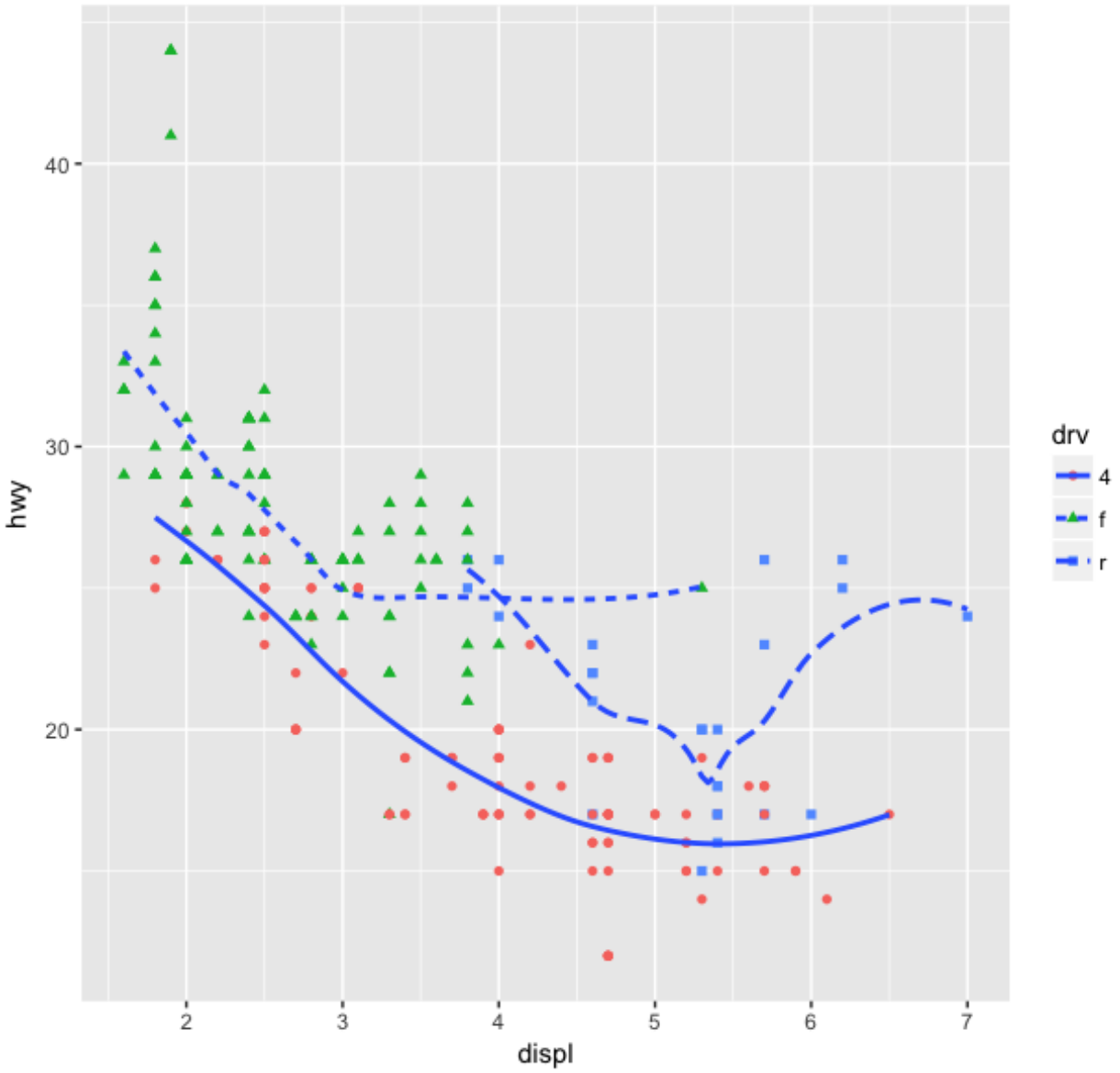
```
ggplot(data = mpg, mapping = aes(x=displ, y = hwy))+
  geom_point(mapping = aes(color=drv,shape=drv)) +
  geom_smooth(se=FALSE)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



1.4.9 Cuestión 4.9.

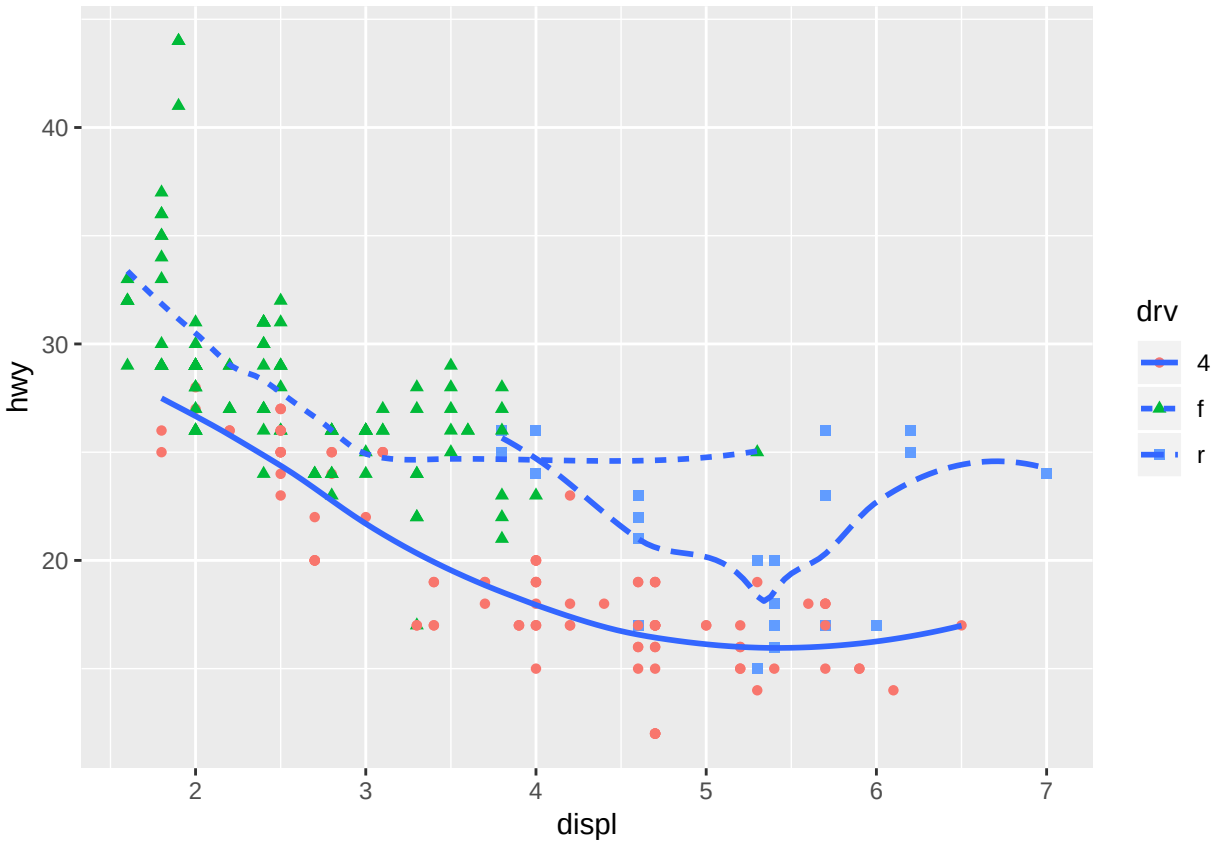
Reproduce el código de R que te genera el siguiente gráfico.



1.4.9.1 Solución

```
ggplot(data = mpg, mapping = aes(x=displ, y = hwy))+
  geom_point(mapping = aes(color=drv,shape=drv)) +
  geom_smooth(mapping = aes(linetype=drv),se=FALSE)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

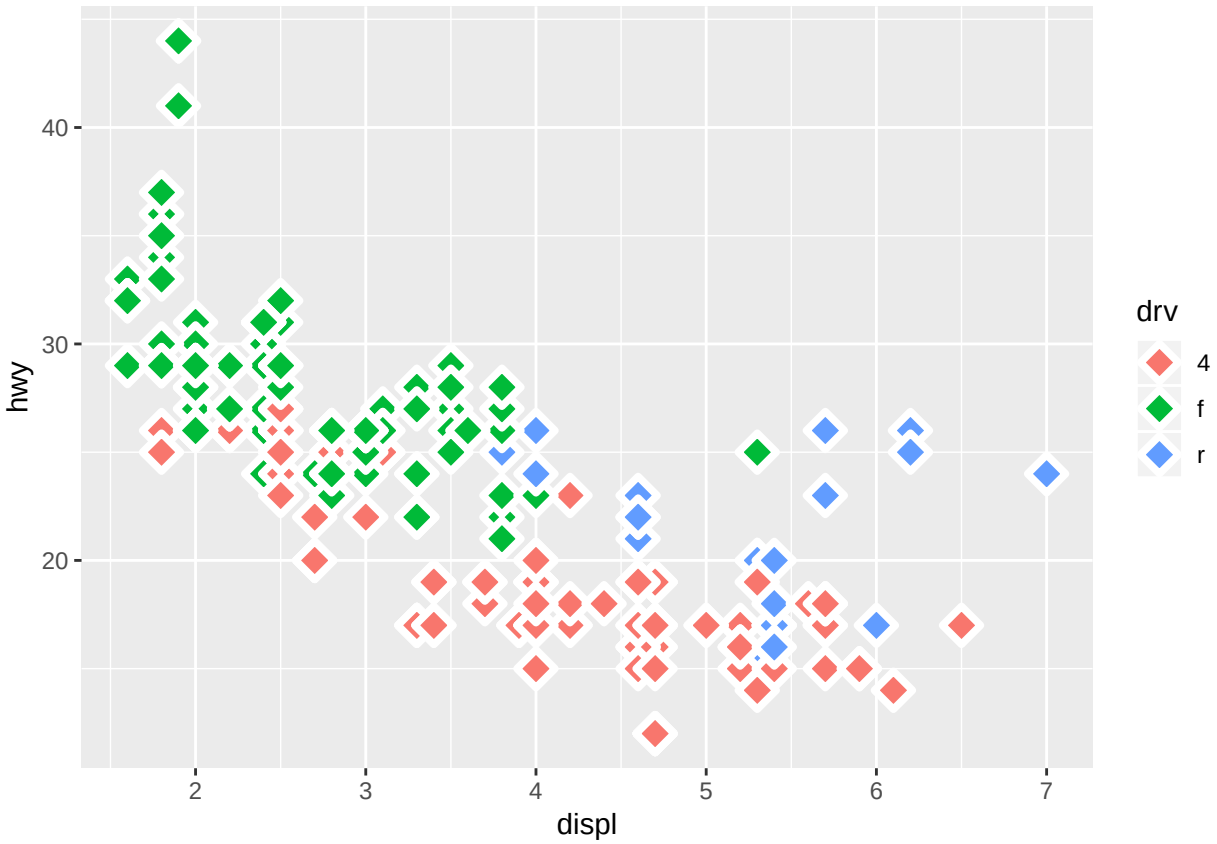


1.4.10 Cuestión 4.10.

Reproduce el código de R que te genera el siguiente gráfico. Investiga algunos parámetros adicionales que te harán falta de ggplot2 como stroke entre otros.

1.4.10.1 Solución

```
ggplot(data = mpg, mapping = aes(x=displ, y = hwy) ) +
  geom_point(mapping = aes(fill = drv), size = 4,
             shape = 23, col = "white", stroke = 2)
```



1.5 Tarea 5: Transformaciones estadísticas ggplot. Sección3: Lecciones 22 y 23

Vamos a usar las transformaciones estadísticas básicas aprendidas. Preguntas de esta tarea

1.5.1 Cuestión 5-1

¿Qué hace el parámetro geom_col? ¿En qué se diferencia de geom_bar?

1.5.1.1 Solución

Según la documentación exacta, os sumará los valores subministrados como y en el dataset, concretamente:

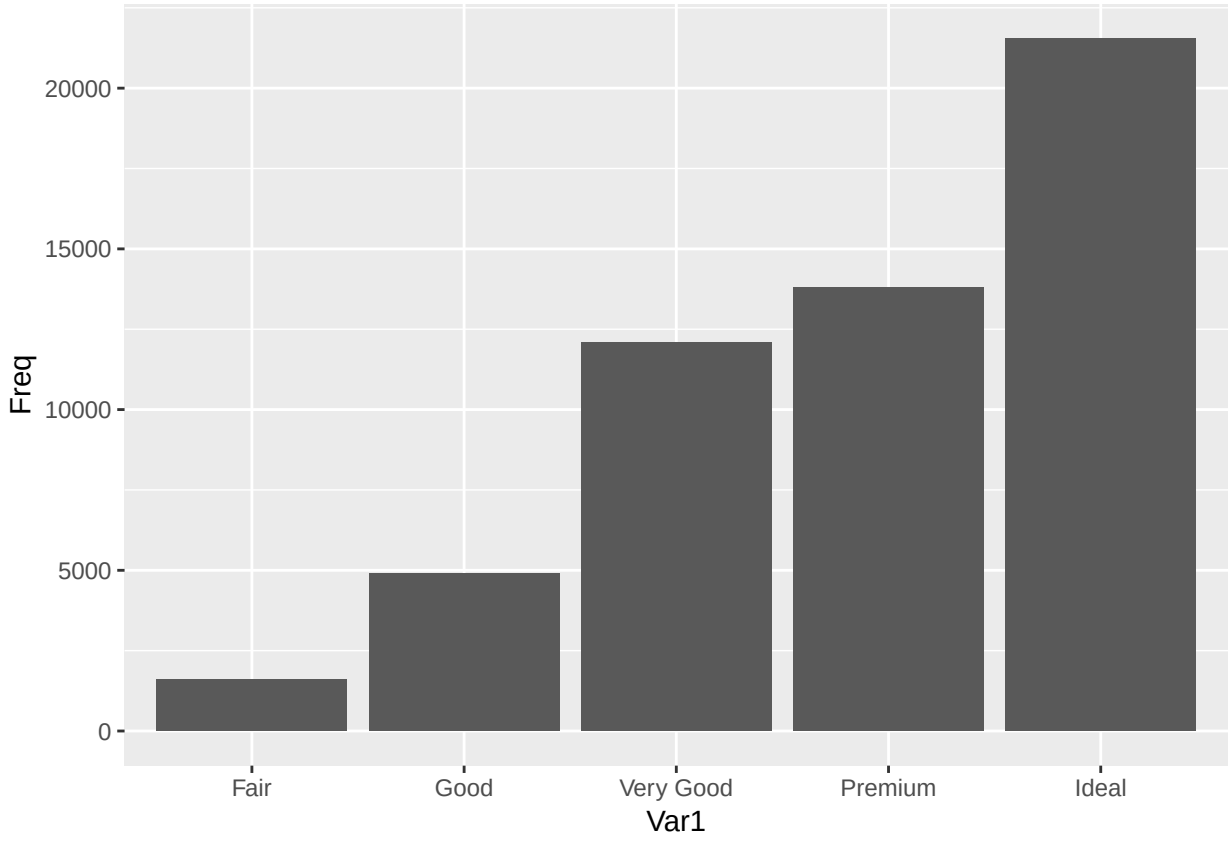
There are two types of bar charts: geom_bar makes the height of the bar proportional to the number of cases in each group (or if the weight aethetic is supplied, the sum of the weights). If you want the heights of the bars to represent values in the data, use geom_col instead. geom_bar uses stat_countby default: it counts the number of cases at each x position. geom_col uses stat_identity: it leaves the data as is.

El siguiente ejemplo ilustra esta situación

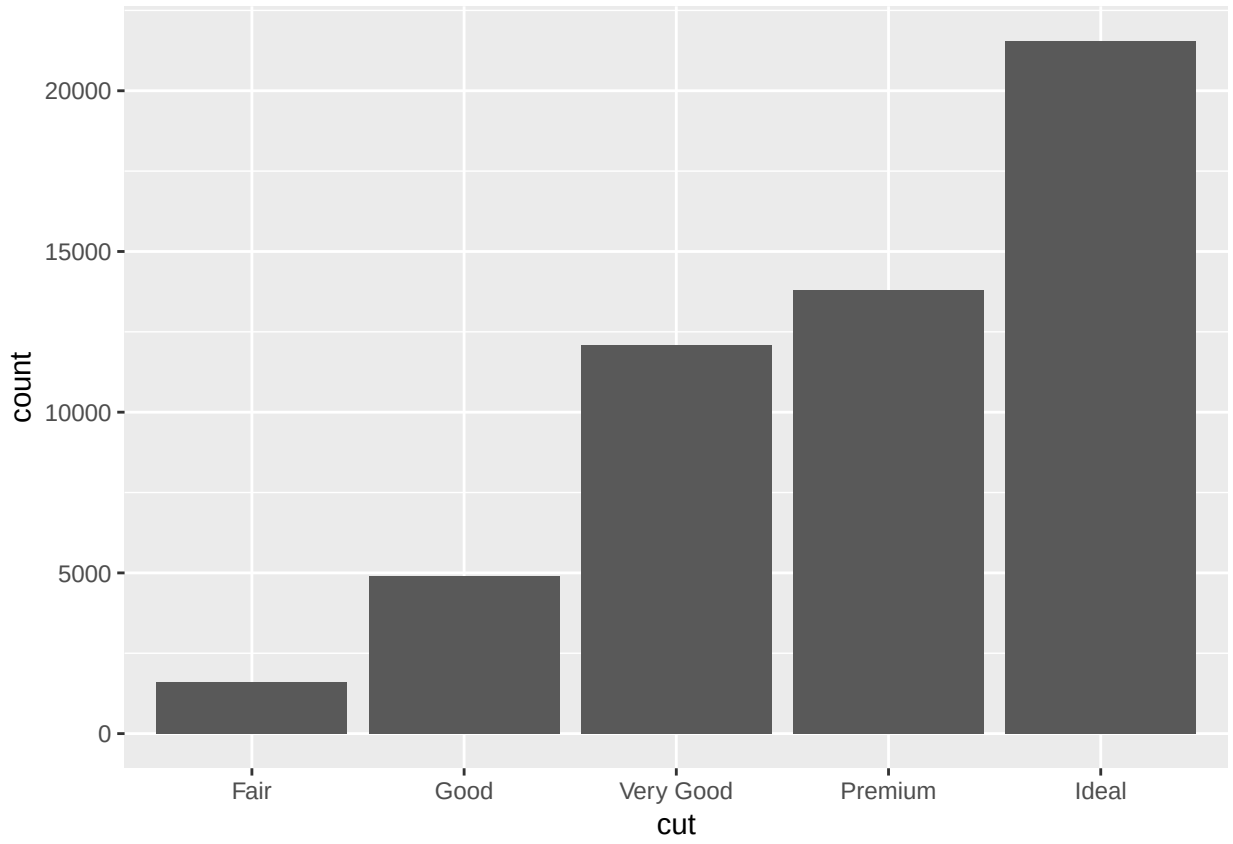
```
df=as.data.frame(table(diamonds$cut))
df
```

```
##        Var1  Freq
## 1      Fair  1610
## 2      Good  4906
## 3 Very Good 12082
## 4   Premium 13791
## 5     Ideal 21551
```

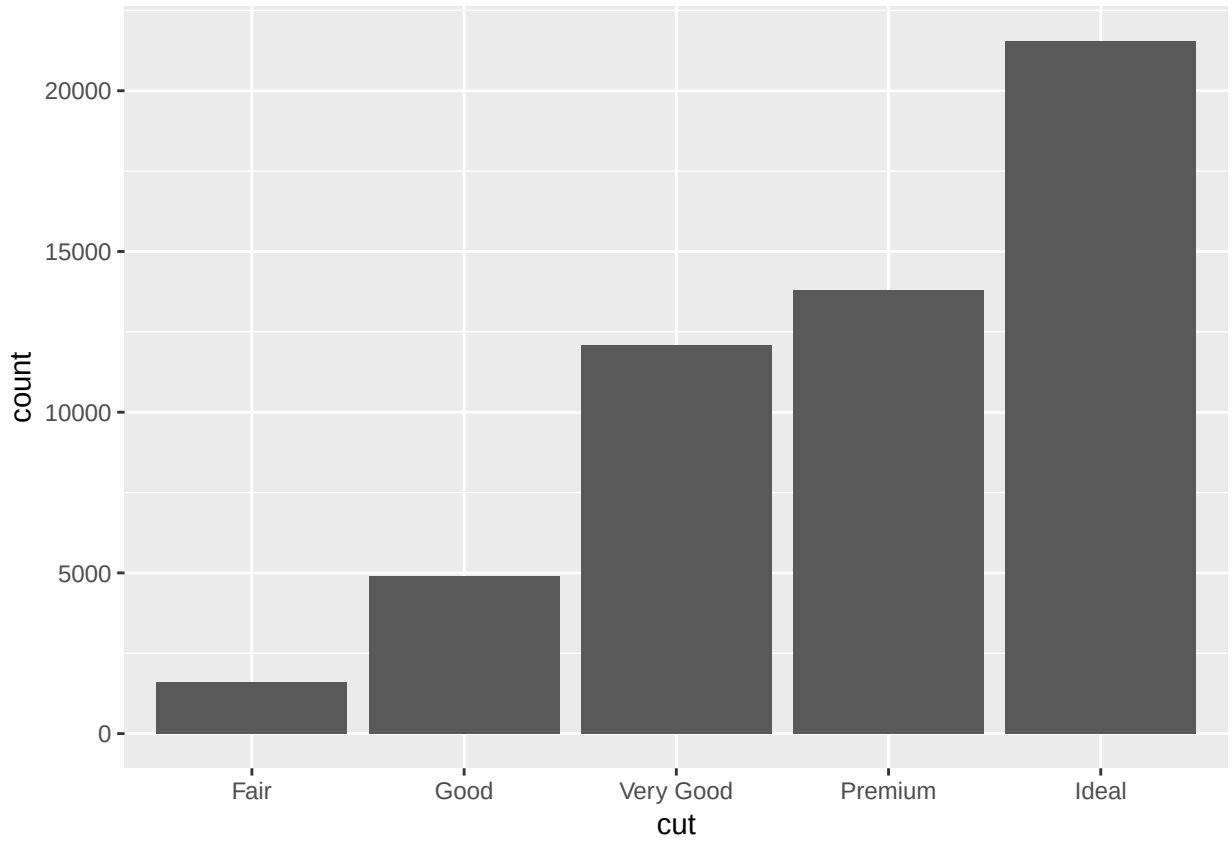
```
ggplot(data = df) +
  geom_col(mapping = aes(x = Var1,y=Freq))
```



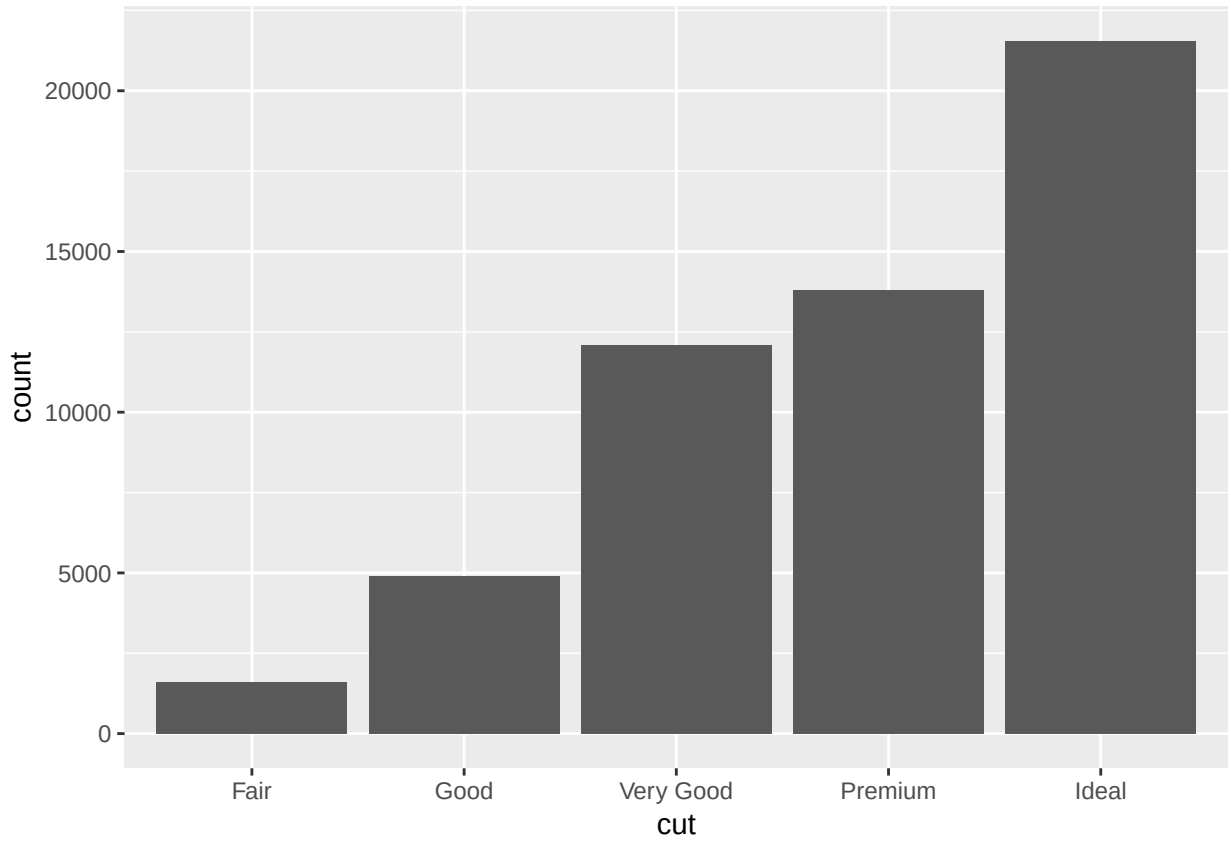

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut,y=..count..))
```



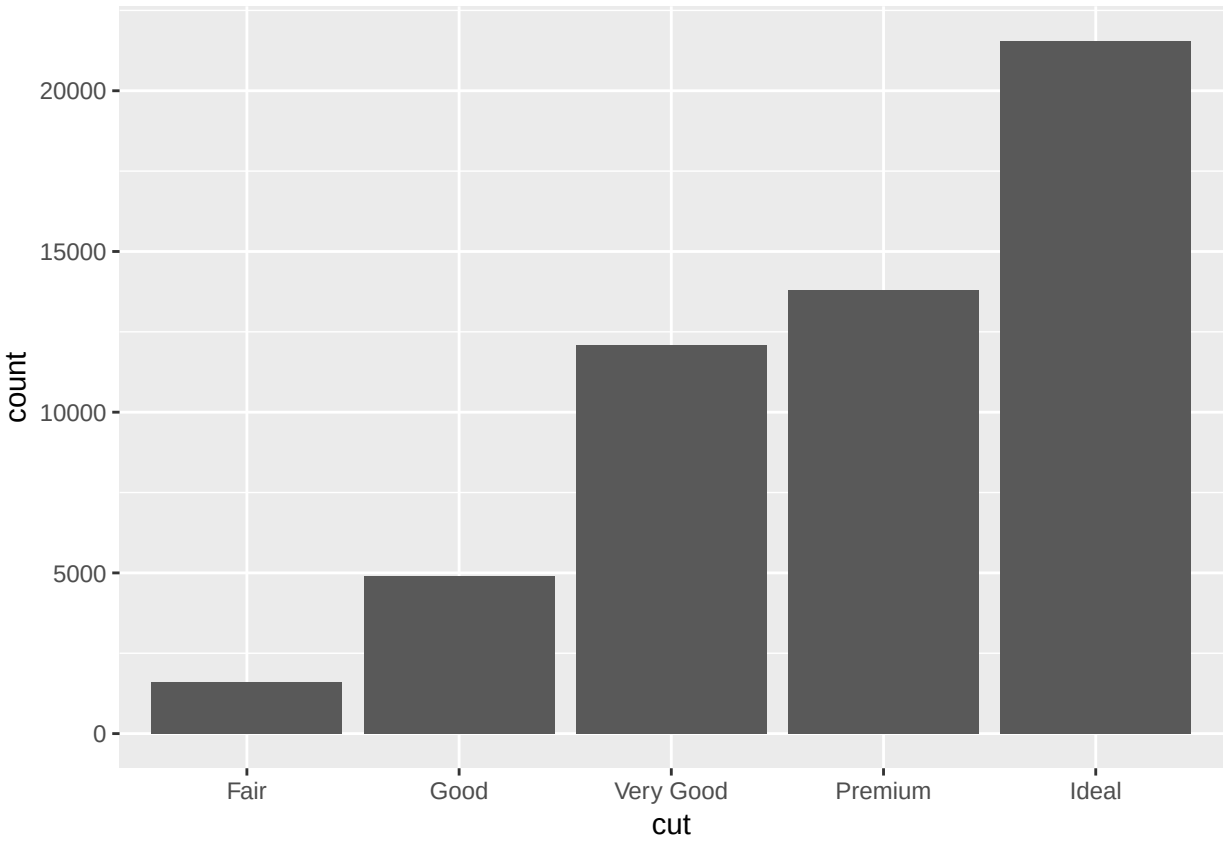

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut),stat="count")
```




```
ggplot(data = diamonds,aes(x = cut)) +
  stat_count()
```




```
ggplot(data = diamonds,aes(x = cut)) +
  stat_count(geom="bar")
```



1.5.2 Cuestión 5.2.

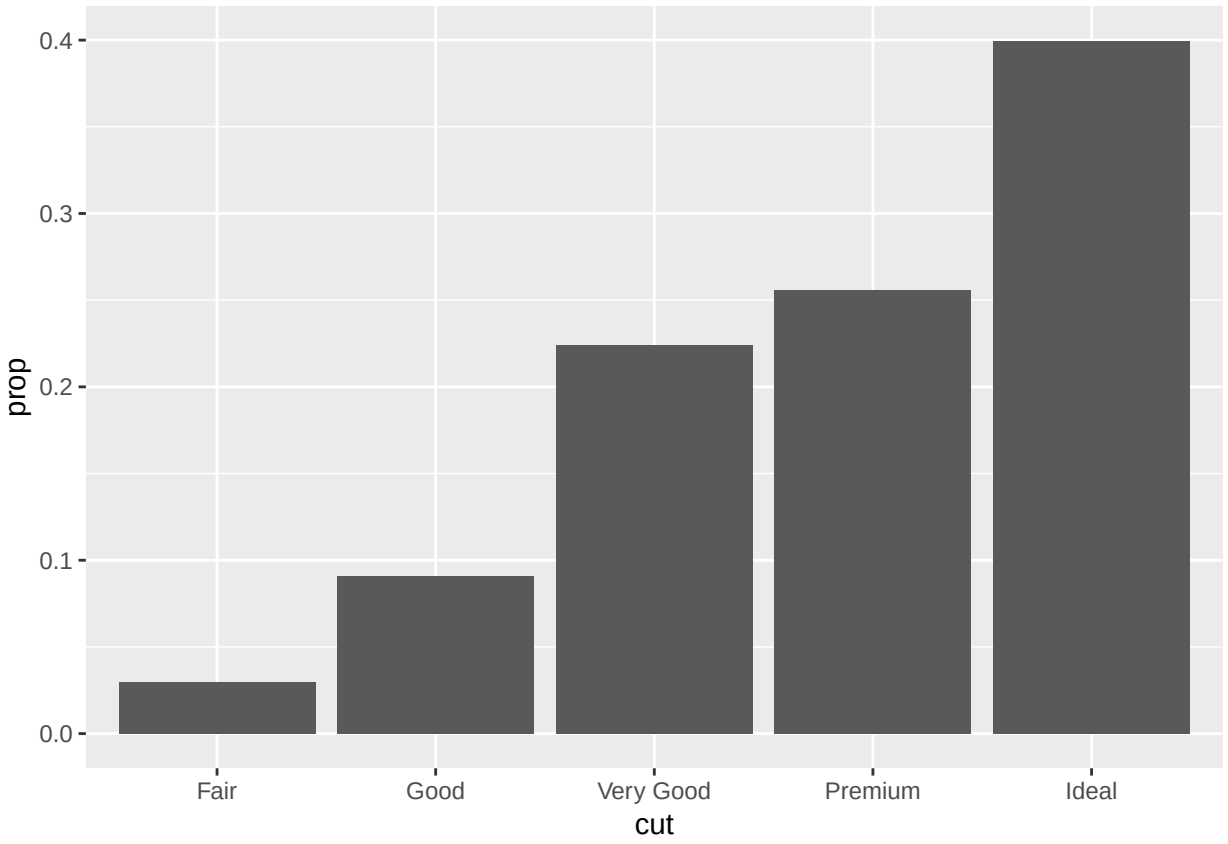
La gran mayoría de geometrías y de stats vienen por parejas que siempre se utilizan en conjunto. Por ejemplo geom_bar con stat_count. Haz una pasada por la documentación y la chuleta de ggplot y establece una relación entre esas parejas de funciones. ¿Qué tienen todas en común?

1.5.2.1 Solución

```
str(diamonds)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    53940 obs. of  10 variables:
##  $ carat  : num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
##  $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
##  $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
##  $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
##  $ depth  : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
##  $ table  : num  55 61 65 58 58 57 57 55 61 61 ...
##  $ price  : int  326 326 327 334 335 336 336 337 337 338 ...
##  $ x      : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ y      : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
##  $ z      : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut,y = ..prop..,group=1))
```



1.5.3 Cuestión 5.3.

¿Qué variables calcula la función stat_smooth? ¿Qué parámetros controlan su comportamiento?

1.5.3.1 Solución

Ver

1.5.4 Cuestión 5.4.

Cuando hemos pintado nuestro diagrama de barras con sus proporciones, necesitamos configurar el parámetro group = 1. ¿Por qué?

1.5.4.1 Solución

Para que calcule las estadisticos agregados para cada niveles de x.

1.5.5 Cuestión 5.5.

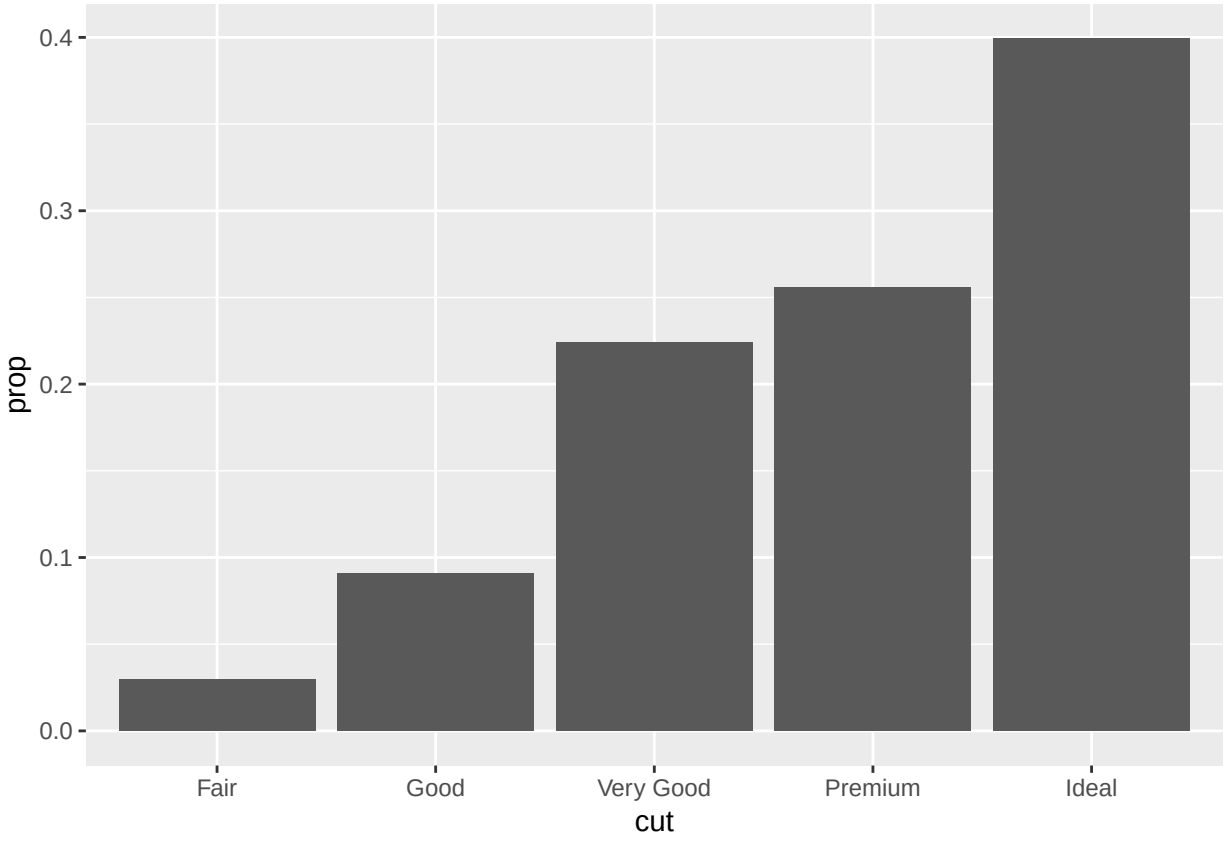
¿Qué problema tienen los dos siguientes gráficos?

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, y = ..prop..))

ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = color, y = ..prop..))
```

1.5.5.1 Solución

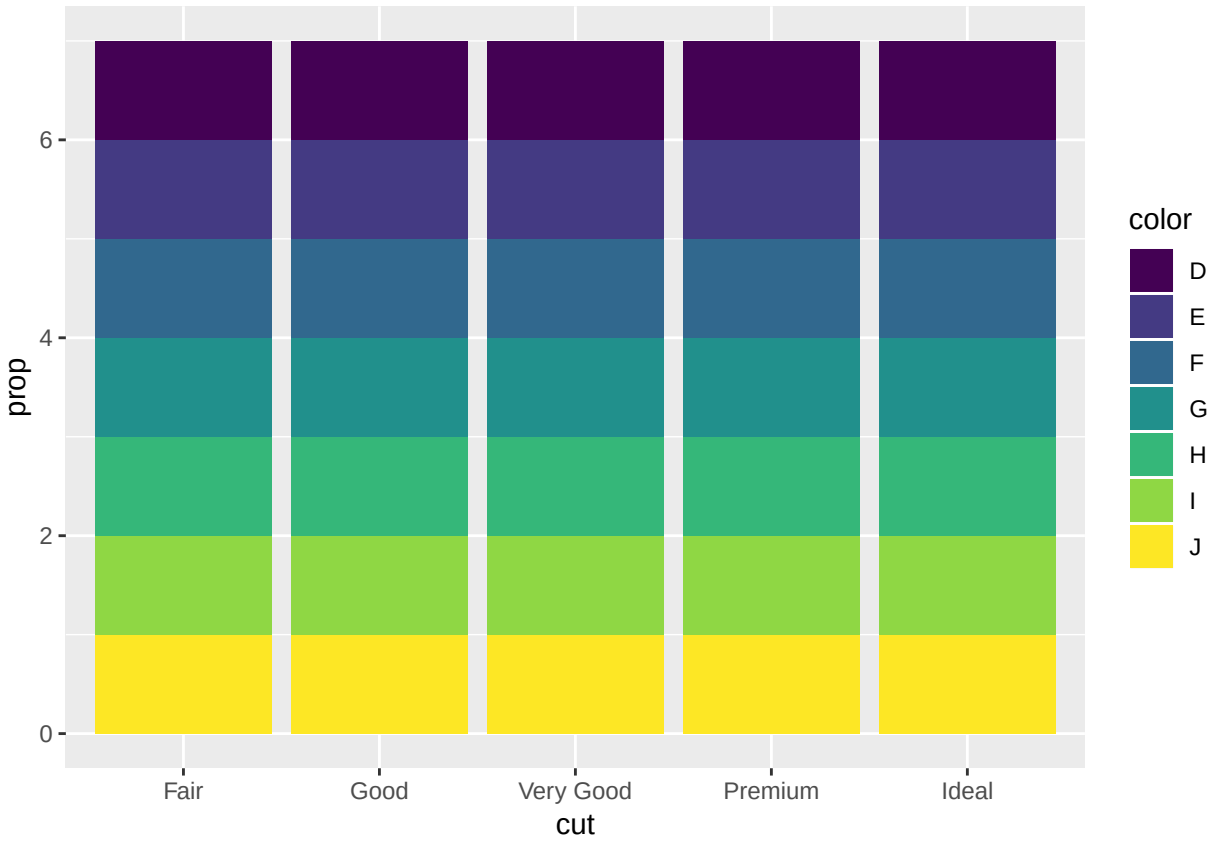
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, y = ..prop..,group=1))
```




```
prop.table(table(diamonds$color,diamonds$cut),2)
```

```
##
##           Fair       Good  Very Good    Premium      Ideal
##   D 0.10124224 0.13493681 0.12522761 0.11623523 0.13150202
##   E 0.13913043 0.19017530 0.19864261 0.16945834 0.18110529
##   F 0.19378882 0.18528333 0.17910942 0.16902328 0.17753237
##   G 0.19503106 0.17753771 0.19028307 0.21202233 0.22662521
##   H 0.18819876 0.14309009 0.15096838 0.17112610 0.14454086
##   I 0.10869565 0.10640033 0.09965238 0.10354579 0.09711846
##   J 0.07391304 0.06257644 0.05611654 0.05858893 0.04157580
```

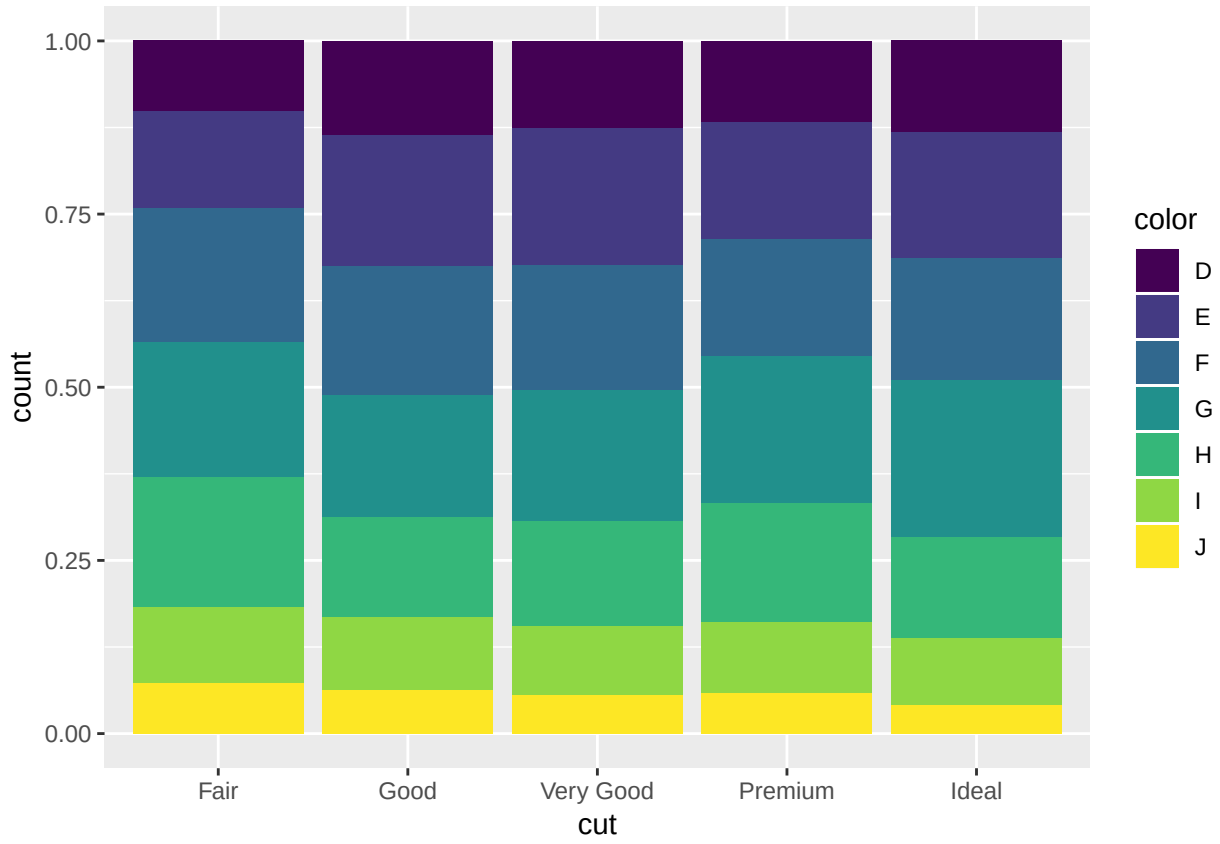
```
ggplot(data = diamonds,mapping = aes(x = cut,  y = ..prop..,fill=color))+
    geom_bar(position="stack")
```



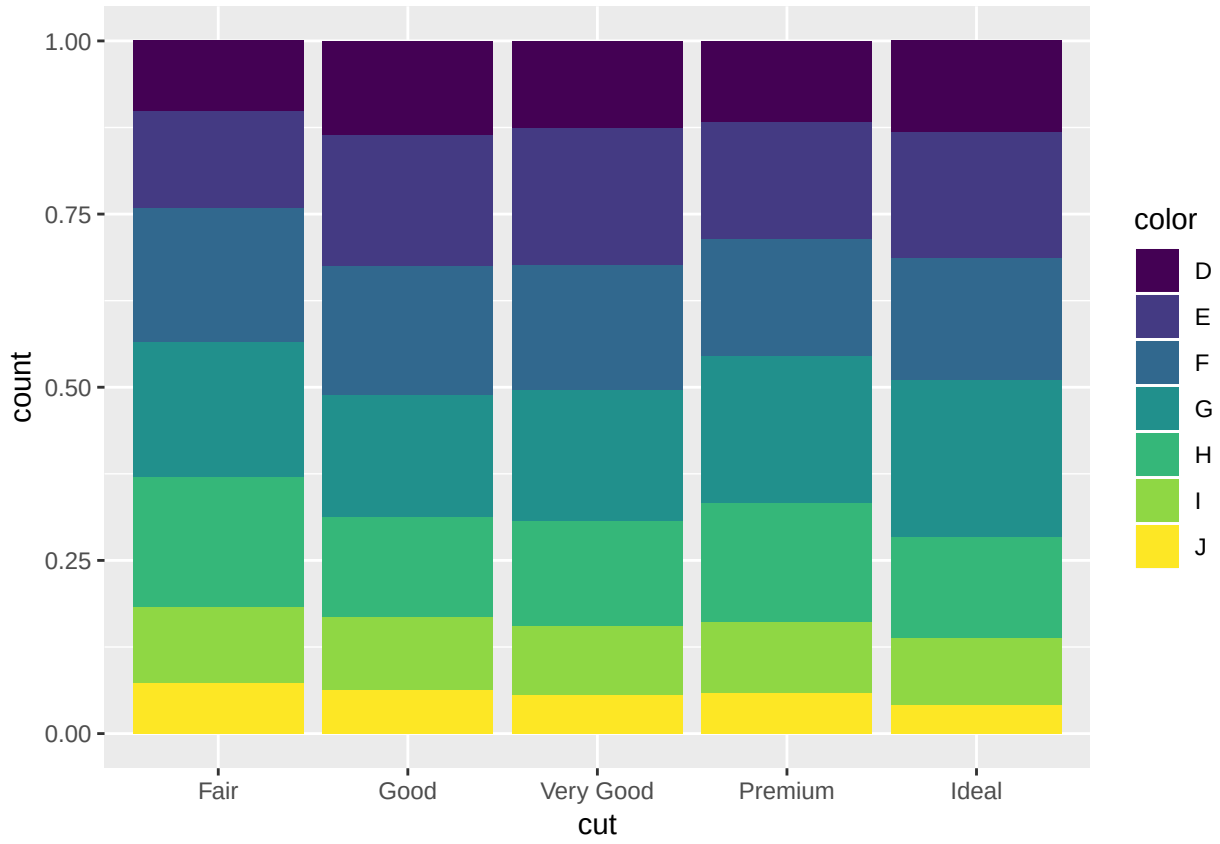
Este útimo `gplot` parece erróneo ver final del documento.

Otras soluciones razonables, y alguna otra incomprensible:

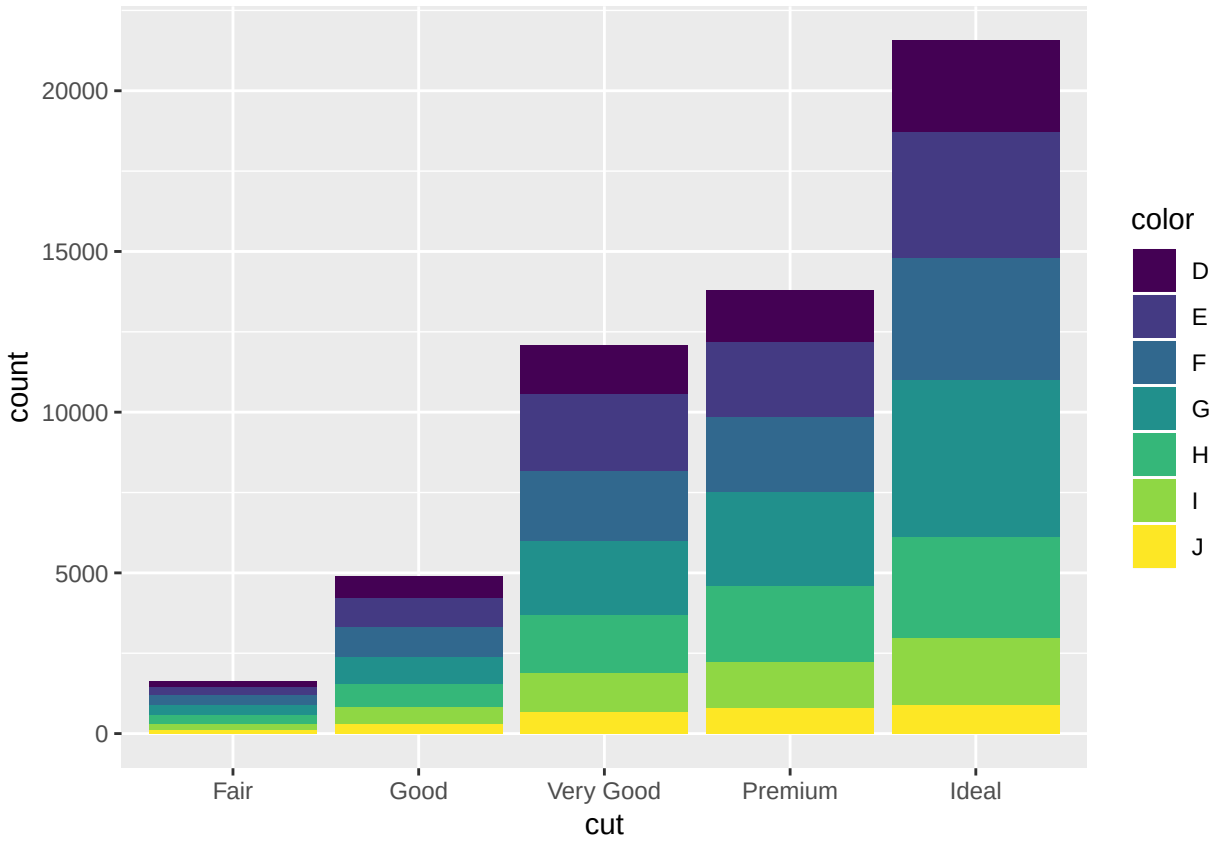
```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = color), position = "fill")
```



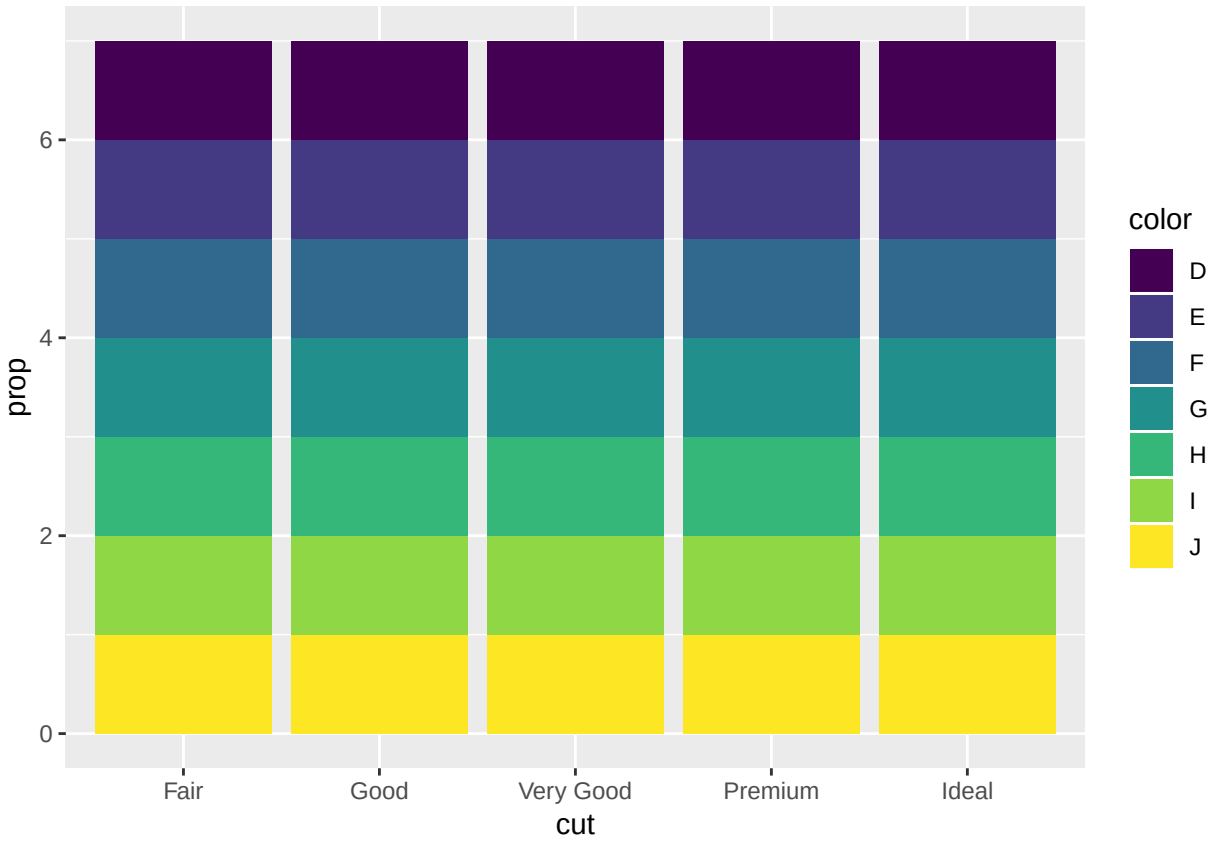

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = color,y=..count..), position = "fill")
```




```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = color,y=..count..), position = "stack")
```



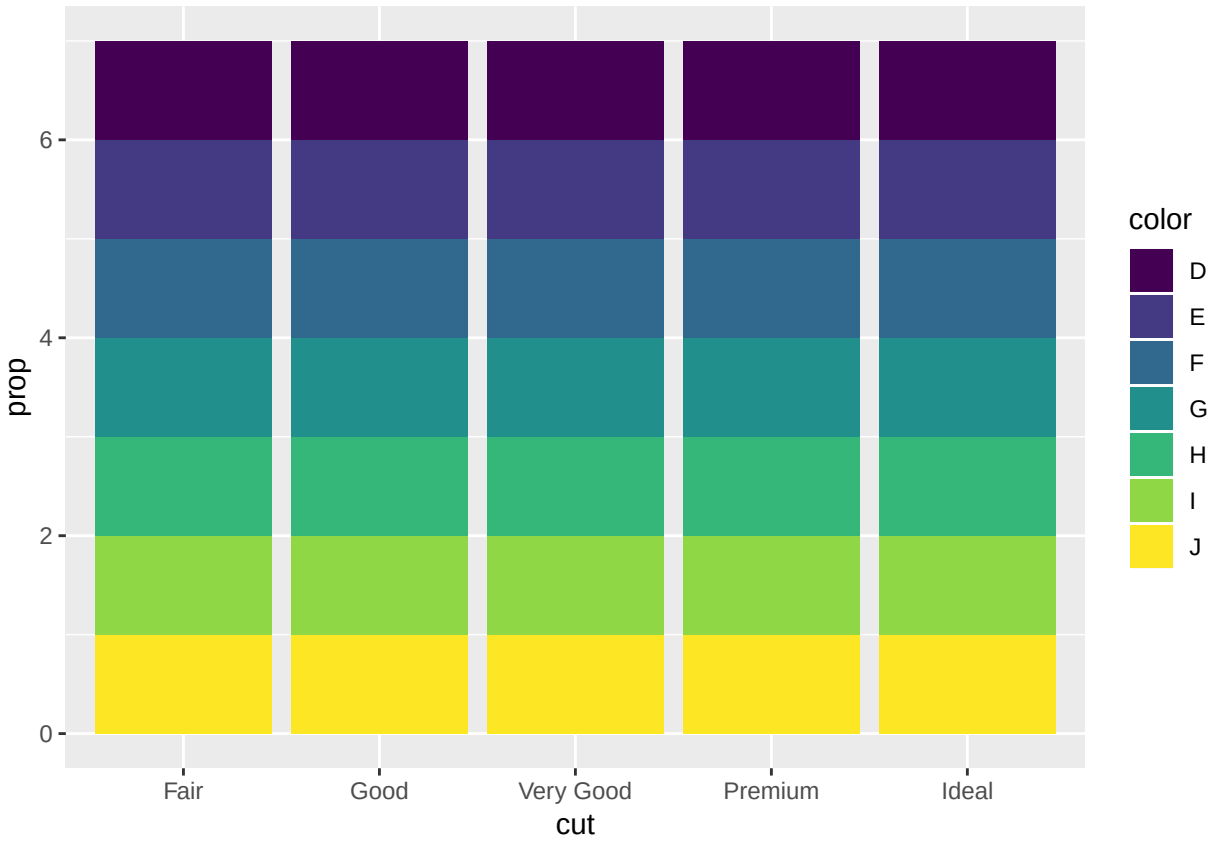

```
### Mal funcionamiento del ggplot2 ?¿?¿? escribir
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = color,y=..prop..), position = "stack")
```



Por último

Volvamos al gráfico

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, fill = color, y = ..prop..))
```



Lo que vemos es:

- Que el eje vertical pone prop miestras que sus valores van de 0 a 6.
- Que todas las cajas del fill parecen de la misma altura, nosotros quería que fueran la proporción de `color` en cada clase de `cut`.

Así que la *graph grammar* de ggplot2 no entiende lo que queremos con esta sintaxis.

1.6 Tarea 6 EJERCICIO. Ajustes avanzados ggplo2 . Sección 3: Lecciones 24 a 27

La lecciones 24 a 27 ecplican ajuestes avanzdos de ggplot. Reolvedlas como ejercicio

1.6.1 Cuestión 6.1.

El siguiente gráfico que genera el código de R es correcto pero puede mejorarse. ¿Qué cosas añadirías para mejorarlo?

```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy )) +
  geom_point()
```

1.6.2 Cuestión 6.2.

Investiga la documentación de geom_jitter(). ¿Qué parámetros controlan la cantidad de ruído aleatorio (jitter)?

1.6.3 Cuestión 6.3.

Compara las funciones geom_jitter contra geom_count y busca semejanzas y diferencias entre ambas.

1.6.4 Cuestión 6.4.

¿Cual es el valor por defecto del parámetro position de un geom_boxplot? Usa el dataset de diamonds o de mpg para hacer una visualización que lo demuestre.

##3 Cuestión 6.5 Convierte un diagrama de barras apilado en un diagrama de sectores o de tarta usando la función coord_polar()

1.6.5 Cuestión 6.6.

¿Qué hace la función labs()? Lee la documentación y explícalo correctamente.

1.6.6 Cuestión 6.7.

¿En qué se diferencian las funciones coord_quickmap() y coord_map()?

1.6.7 Cuestión 6.8.

Investiga las coordenadas coord_fixed() e indica su función.

1.6.8 Cuestión 6.9.

Investiga la geometría de la función geom_abline(), geom_vline() y geom_hline() e indica su función respectivamente.

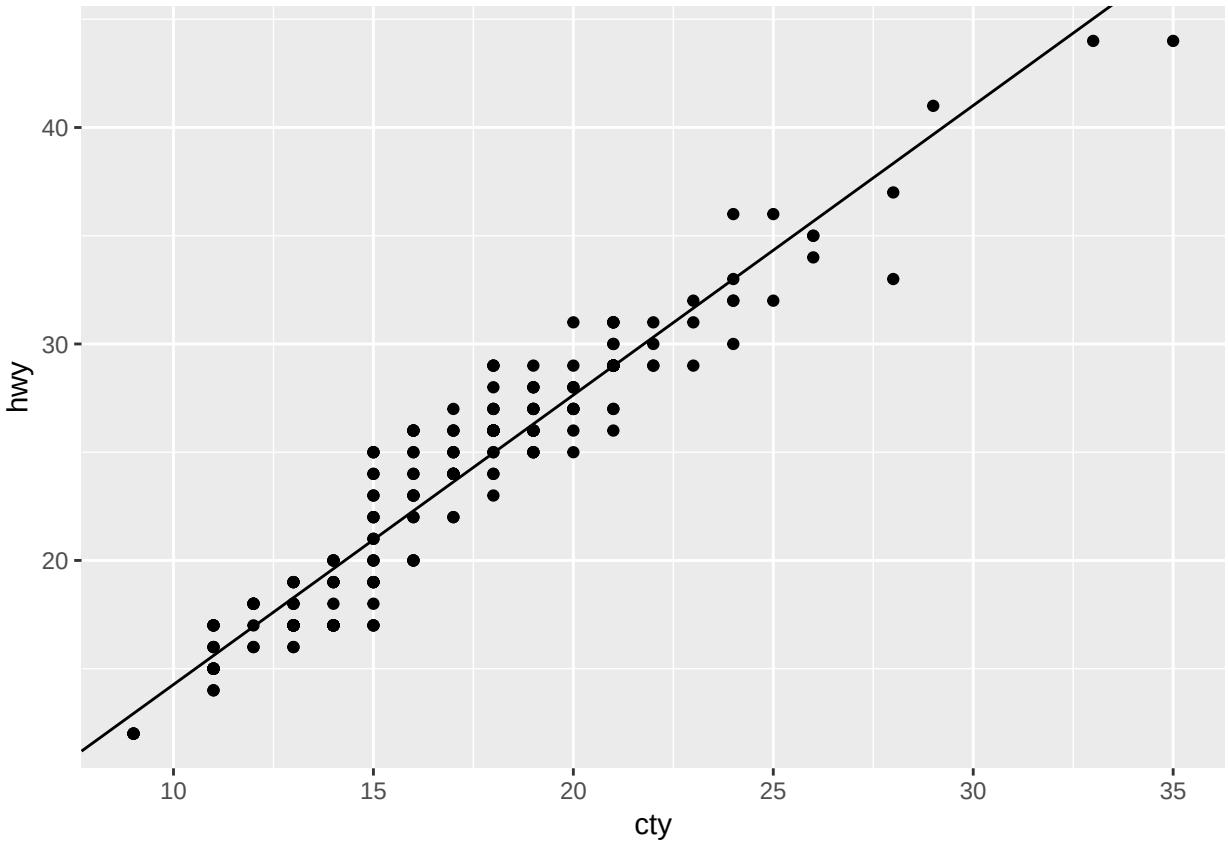
1.6.9 Cuestión 6.10.

¿Qué nos indica el gráfico siguiente acerca de la relación entre el consumo en ciudad y en autopista del dataset de mpg?

```
coef=lm(mpg$hwy~mpg$cty)$coefficients
coef
```

```
## (Intercept)     mpg$cty
##   0.8920411   1.3374556
```

```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy )) +
  geom_point() +
  geom_abline(slope=coef[2],intercept=coef[1])
```

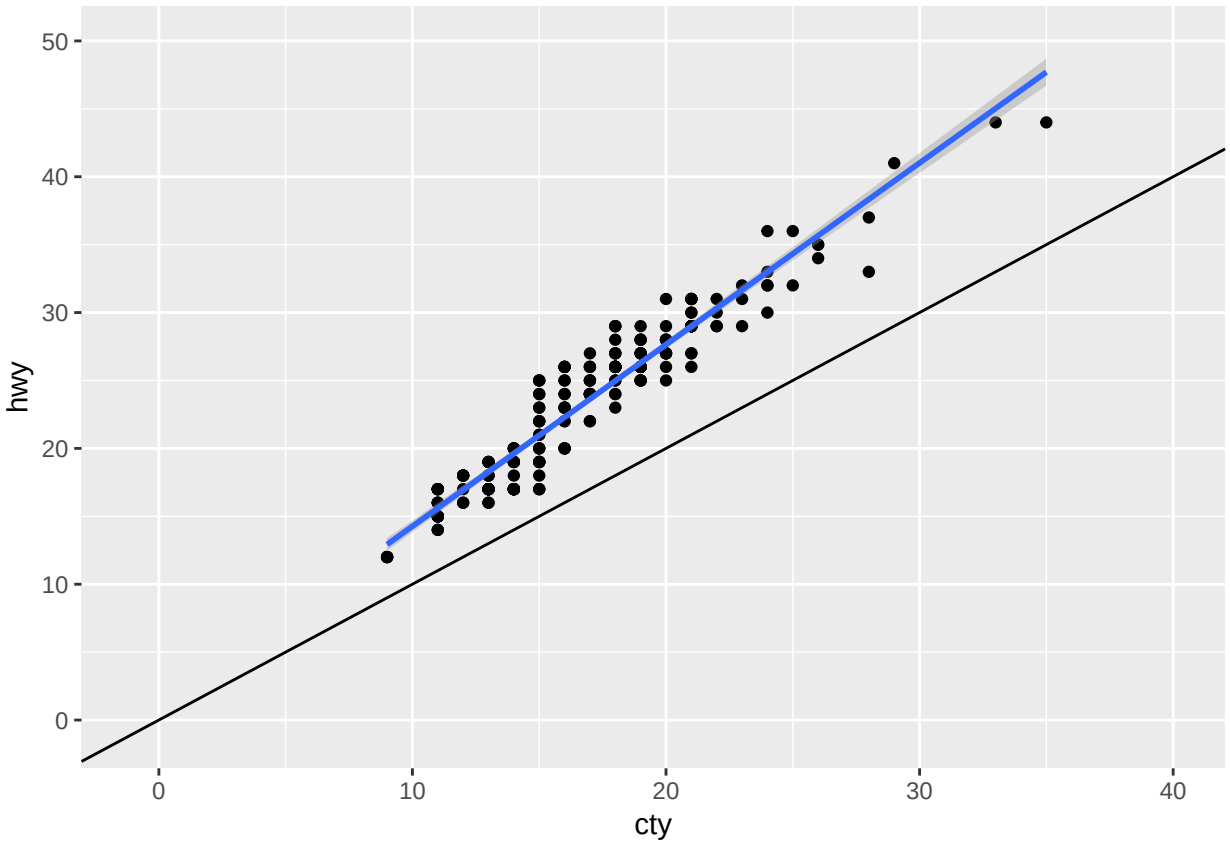


1.6.9.1 Solución

Notemos que en la versión original del libro añadía geom_abline() sin prámetros, esto No es correcto como se ve el el sigueinte código. Comentaremos en clase el código.

```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy )) +
  geom_point() +
  geom_smooth(method="lm")+
  coord_fixed()+
  coord_cartesian(xlim=c(-1,40),ylim=c(-1,50))+
  geom_abline()
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```

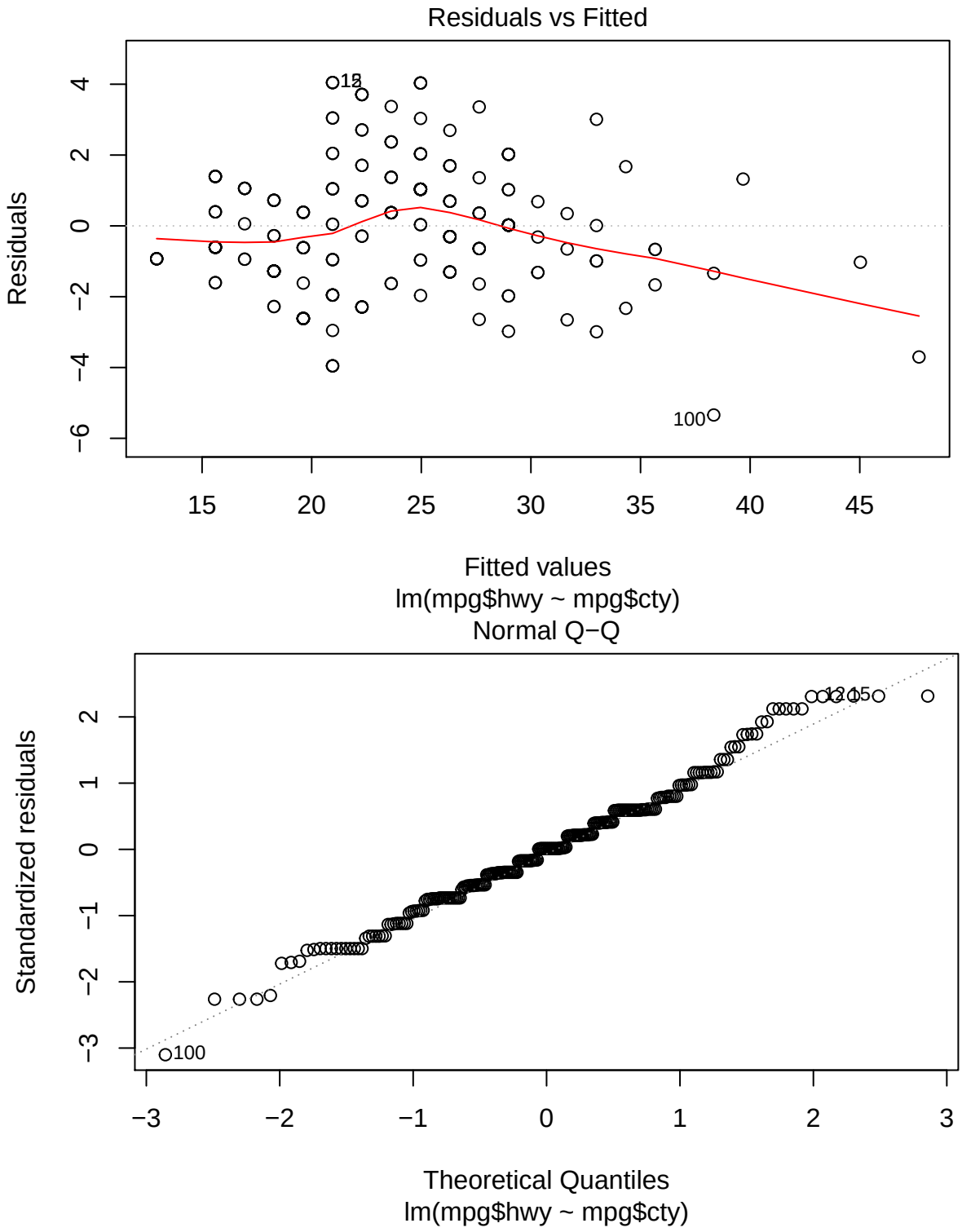


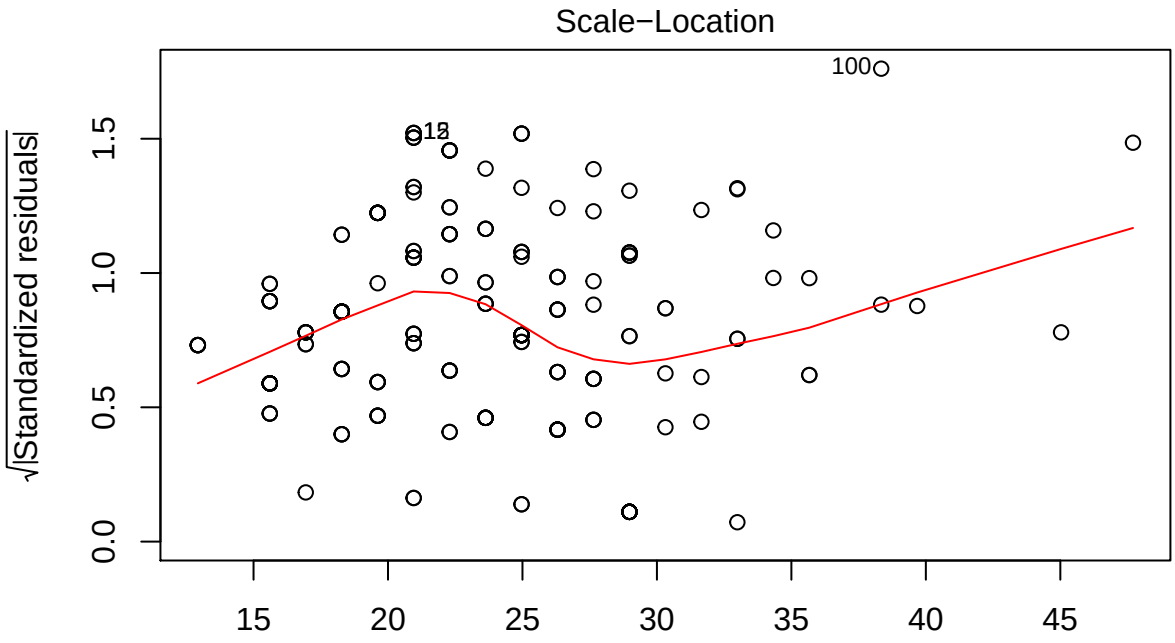
Para evaluar este modelo utilizad el manual Practical Regression and Anova using R de de [Julian [Julian J. Faraway] (http://www.maths.bath.ac.uk/~jjf23/) .

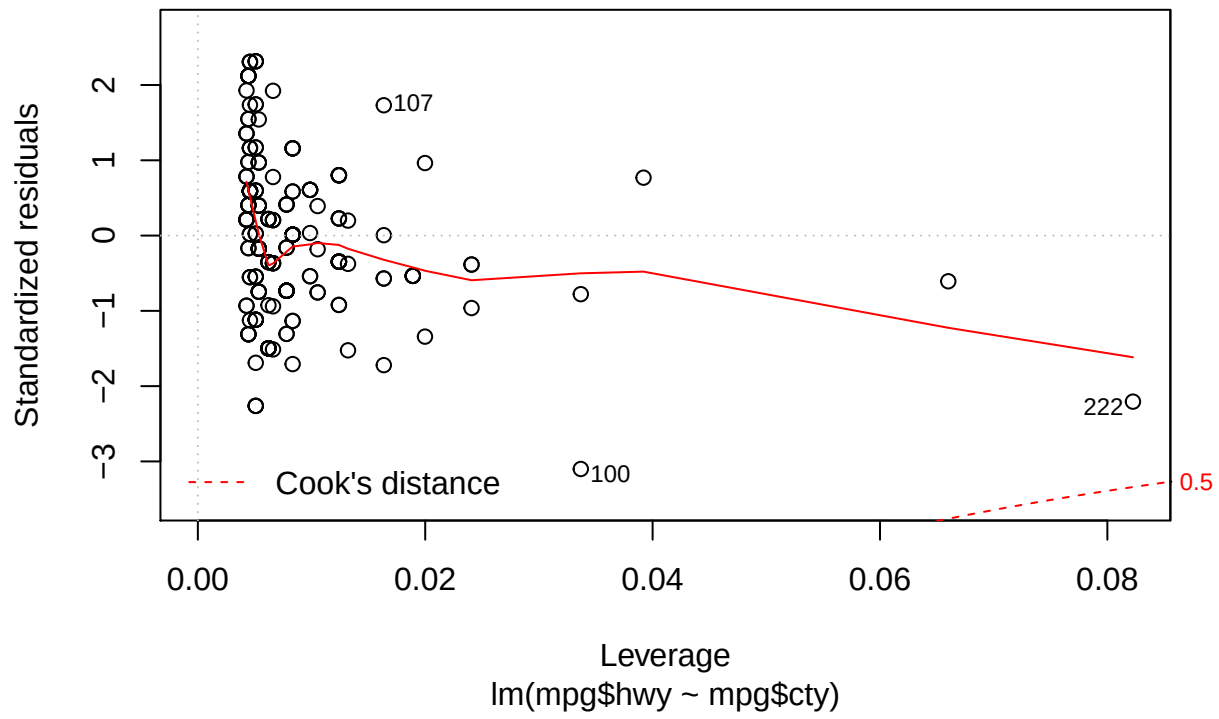
```
lm_model=lm(mpg$hwy~mpg$cty)
summary(lm_model)
```

```
##
## Call:
## lm(formula = mpg$hwy ~ mpg$cty)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.3408 -1.2790  0.0214  1.0338  4.0461
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.89204    0.46895   1.902   0.0584 .
## mpg$cty      1.33746    0.02697  49.585   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.752 on 232 degrees of freedom
## Multiple R-squared:  0.9138, Adjusted R-squared:  0.9134
## F-statistic:  2459 on 1 and 232 DF,  p-value: < 2.2e-16
```

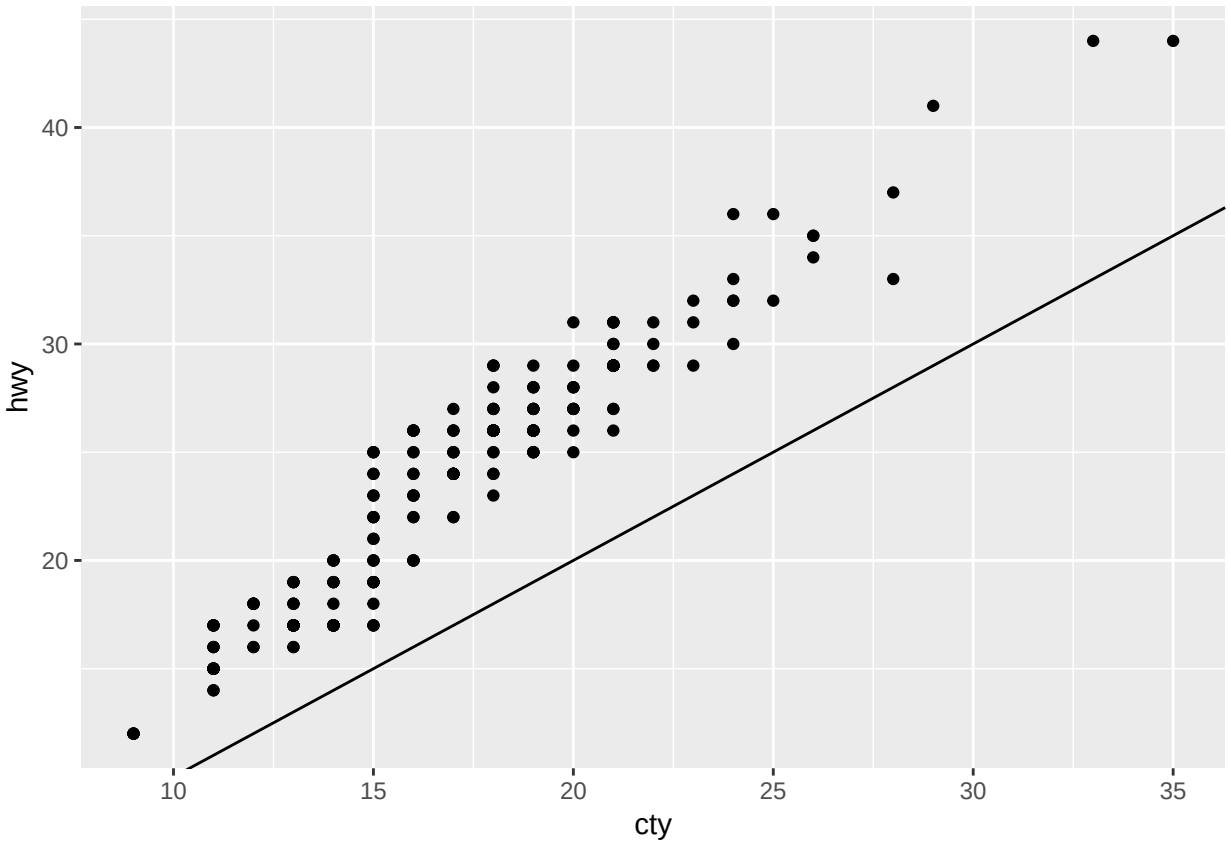
```
plot(lm_model)
```


Residuals vs Fitted
Residuals
Fitted values
lm(mpg$hwy ~ mpg$cty)
Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(mpg$hwy ~ mpg$cty)






```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy )) +
    geom_point() +
    geom_abline()
```



2 Tareas sección 4: Lecciones 28 a 34

Estas lecciones no necesitan taller. Consisten e n explicar algunos aspecto generalers de R y Rstudio desde la alculadora a los proyectos de Rstudio. La hemos comentado en la clase del martés 17 de septiembre y en anteriores cursos de GMAT.

De todas maneras dejamos las preguntas de la calculadora y Rstudio básico como ejercicio.

2.1 Tarea 7: Introducción a R como herramienta de cálculo. Lecciones 28 a 31.

2.1.1 Cuestión 7.1.

¿Por qué no funciona el siguiente código?

```
my_variable <- 5
my.variable
```

2.1.2 Cuestión 7.2.

¿Por qué no funciona el siguiente código?

```
my_variable <- 5
my_varIable
```

2.1.3 Cuestión 7.3.

¿Por qué no funciona el siguiente código?

```
my_variable <- 5
```

```
my_var1able
```

2.1.4 Cuestión 7.4.

**¿Por qué no funciona el siguiente código?

```
my_variable -> 5
my_variable
```

2.1.5 Cuestión 7.5.

Las siguientes líneas pueden tener algun error de escritura. Localízalo y corrígelo para que funcione correctamente.

```
librari(tidyverse)
ggplot(dati = mpg) + geom_puint(mapping = aes(x = displ, y = hwy))
fliter(mpg, cyl=6)
filter(diamond, caret > 4)
```

2.1.6 Cuestión 7.6.

Vamos a por un poco de magia oscura. Prueba la combinación Alt + Shift + K. ¿Qué hace? ¿Útil eh?

3 Tareas Sección 5: La transformación de los datos. Lecciones 35 a 52

3.1 Tarea 8: Filyrando datos con dplyr. Sección 5: Lecciones 35 a 39

Preguntas de esta tarea. El objetivo es que des las instrucciones precisas de `dplyr` que nos dan los vuelos con las condiciones que se indiquen en cada ejercicio.

3.1.1 Cuestión 8.1.

Encuentra todos los vuelos que llegaron más de una hora tarde de lo previsto.

3.1.1.1 Solución

```
library(nycflights13)
filter(flights, arr_delay>60)
```

```
## # A tibble: 27,789 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      811            630       101     1047
##  2  2013     1     1      848           1835       853     1001
##  3  2013     1     1      957            733       144     1056
##  4  2013     1     1     1114            900       134     1447
##  5  2013     1     1     1120            944        96     1331
##  6  2013     1     1     1255           1200        55     1451
##  7  2013     1     1     1301           1150        71     1518
##  8  2013     1     1     1337           1220        77     1649
##  9  2013     1     1     1342           1320        22     1617
## 10  2013     1     1     1400           1250        70     1645
## # ... with 27,779 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
```

```
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

3.1.2 Cuestión 8.2.

Encuentra todos los vuelos que volaron hacia San Francisco (aeropuertos SFO y OAK)

3.1.2.1 Solución

```
filter(flights, dest == "SFO" | dest == "OAK")
```

```
## # A tibble: 13,643 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      558            600        -2      923
##  2  2013     1     1      611            600        11      945
##  3  2013     1     1      655            700        -5     1037
##  4  2013     1     1      729            730        -1     1049
##  5  2013     1     1      734            737        -3     1047
##  6  2013     1     1      745            745         0     1135
##  7  2013     1     1      746            746         0     1119
##  8  2013     1     1      803            800         3     1132
##  9  2013     1     1      826            817         9     1145
## 10  2013     1     1     1029           1030        -1     1427
## # ... with 13,633 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

3.1.3 Cuestión 8.3.

Encuentra todos los vuelos operados por United American (UA) o por American Airlines (AA)

3.1.3.1 Solución

```
filter(flights, carrier == "UA" | carrier == "AA")
```

```
## # A tibble: 91,394 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      554            558        -4      740
##  5  2013     1     1      558            600        -2      753
##  6  2013     1     1      558            600        -2      924
##  7  2013     1     1      558            600        -2      923
##  8  2013     1     1      559            600        -1      941
##  9  2013     1     1      559            600        -1      854
## 10  2013     1     1      606            610        -4      858
## # ... with 91,384 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

3.1.4 Cuestión 8.4.

Encuentra todos los vuelos que salieron los meses de primavera (Abril, Mayo y Junio)

3.1.4.1 Solución

```
filter(flights, month %in% c(4,5,6))
```

```
## # A tibble: 85,369 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     4     1      454            500        -6      636
##  2  2013     4     1      509            515        -6      743
##  3  2013     4     1      526            530        -4      812
##  4  2013     4     1      534            540        -6      833
##  5  2013     4     1      542            545        -3      914
##  6  2013     4     1      543            545        -2      921
##  7  2013     4     1      551            600        -9      748
##  8  2013     4     1      552            600        -8      641
##  9  2013     4     1      553            600        -7      725
## 10  2013     4     1      554            600        -6      752
## # ... with 85,359 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

3.1.5 Cuestión 8.5.

Encuentra todos los vuelos que llegaron más de una hora tarde pero salieron con menos de una hora de retraso.

3.1.5.1 Solución

```
filter(flights, arr_delay > 60, dep_delay <= 60)
```

```
## # A tibble: 5,124 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1     1255           1200        55     1451
##  2  2013     1     1     1342           1320        22     1617
##  3  2013     1     1     1402           1323        39     1650
##  4  2013     1     1     1411           1315        56     1717
##  5  2013     1     1     1424           1349        35     1701
##  6  2013     1     1     1428           1329        59     1803
##  7  2013     1     1     1558           1534        24     1808
##  8  2013     1     1     1604           1510        54     1817
##  9  2013     1     1     1608           1535        33     2002
## 10  2013     1     1     1630           1548        42     1902
## # ... with 5,114 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

3.1.6 Cuestión 8.6.

Encuentra todos los vuelos que salieron con más de una hora de retraso pero consiguieron llegar con menos de 30 minutos de retraso (el avión aceleró en el aire)

3.1.6.1 Solución

```
filter(flights, arr_delay > 60, dep_delay <= 30)
```

```
## # A tibble: 1,986 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1     1342           1320        22     1617
##  2  2013     1     1     1558           1534        24     1808
##  3  2013     1     1     1751           1745         6     2015
##  4  2013     1     1     2000           1930        30     2255
##  5  2013     1     2      841            845        -4     1134
##  6  2013     1     2      928            905        23     1331
##  7  2013     1     2     1558           1600        -2     1923
##  8  2013     1     6      654            655        -1     1025
##  9  2013     1     6      906            904         2     1313
## 10  2013     1     6     1932           1910        22     2318
## # ... with 1,976 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

3.1.7 Cuestión 8.7.

Encuentra todos los vuelos que salen entre medianoche y las 7 de la mañana (vuelos nocturnos).

3.1.7.1 Solución

```
filter(flights, hour >= 0, hour < 7)
```

```
## # A tibble: 27,905 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # ... with 27,895 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

3.1.8 Cuestión 8.8.

Investiga el uso de la función `between()` de `dplyr`. ¿Qué hace? Puedes usarlo para resolver la sintaxis necesaria para responder alguna de las preguntas anteriores?

3.1.8.1 Solución

```
filter(flights, between(hour,0,6))
```

```
## # A tibble: 27,905 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # ... with 27,895 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

3.1.9 Cuestión 8.9.

¿Cuantos vuelos tienen un valor desconocido de `dep_time`?

3.1.9.1 Solución

```
filter(flights, is.na(dep_time))
```

```
## # A tibble: 8,255 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1       NA           1630        NA       NA
##  2  2013     1     1       NA           1935        NA       NA
##  3  2013     1     1       NA           1500        NA       NA
##  4  2013     1     1       NA            600        NA       NA
##  5  2013     1     2       NA           1540        NA       NA
##  6  2013     1     2       NA           1620        NA       NA
##  7  2013     1     2       NA           1355        NA       NA
##  8  2013     1     2       NA           1420        NA       NA
##  9  2013     1     2       NA           1321        NA       NA
## 10  2013     1     2       NA           1545        NA       NA
## # ... with 8,245 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Podrían haber (en este dataset no) NULLs u otros desconocidos dependiendo cómo se han codificado estos valores.

3.1.10 Cuestión 8.10.

¿Qué variables del dataset contienen valores desconocidos? ¿Qué representan esas filas donde faltan los datos?

3.1.10.1 Solución

Todo lo que sigue son suposiciones, siempre hay que comprobar el motivo de tener datos raros. Los campos desconocidos son fechas de salida/llegada así como retraso de salida/llegada. Puede que se trate de vuelos cancelados, sobretodo por las cifras que manejamos (unos 8000 vuelos anuales).

3.1.11 Cuestión 8.11.

Ahora vas a sorprenderte con la magia oscura... Contesta que dan las siguientes condiciones booleanas

```
NA^0
NA|TRUE
FALSE&NA
```

Intenta establecer la regla general para saber cuando es o no es `NA` (cuidado con `NA*0`)

3.1.11.1 Solución

NA no es un número es NA.... AH pero es un `logical`, no sé el motivo de que de 1 cuando hacemos `NA^0` resultado. Las demás instrucciones sí son bastante "lógicas" un `OR` con un TRUE es siempre TRUE independientemente de que se desconozca la otra entrada del OR, es similar el comportamiento con FALSE

```
class(NA)
```

```
## [1] "logical"
```

```
str(NA)
```

```
##  logi NA
```

```
mode(NA)
```

```
## [1] "logical"
```

```
typeof(NA)
```

```
## [1] "logical"
```

Cuidado!!!! cosas que pasan con los lenguajes que no controlan los tipos de datos.

```
0^0
```

```
## [1] 1
```

```
FALSE^0
```

```
## [1] 1
```

```
TRUE^0
```

```
## [1] 1
```

```
NA^0
```

```
## [1] 1
```

```
NA^1
```

```
## [1] NA
```

3.2 Tarea 9: Ordenación y selección de datos con dplyr. Lecciones 41 y 42.

Repasa los vídeos sobre las funciones `arrange` y `select` de `dplyr` para comprobar que has entendido como funcionan. Preguntas de esta tarea

3.2.1 Cuestión 9.1.

Piensa cómo podrías usar la función `arrange()` para colocar todos los valores NA al inicio. Pista: puedes usar la función `is.na()` en lugar de la función `desc()` como argumento de `arrange`.

3.2.1.1 Solución

```
arrange(flights,!is.na(dep_time))
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1       NA           1630        NA       NA
##  2  2013     1     1       NA           1935        NA       NA
##  3  2013     1     1       NA           1500        NA       NA
##  4  2013     1     1       NA            600        NA       NA
##  5  2013     1     2       NA           1540        NA       NA
##  6  2013     1     2       NA           1620        NA       NA
##  7  2013     1     2       NA           1355        NA       NA
##  8  2013     1     2       NA           1420        NA       NA
##  9  2013     1     2       NA           1321        NA       NA
## 10  2013     1     2       NA           1545        NA       NA
## # ... with 336,766 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

3.2.2 Cuestión 9.2.

Ordena los vuelos de `flights` para encontrar los vuelos más retrasados en la salida. ¿Qué vuelos fueron los que salieron los primeros antes de lo previsto?

3.2.2.1 Solución

```
# El vuelo con mayor retraso fue
arrange(flights, desc(dep_delay))[1:2,] #muestro los dos primero podría haber empates
```

```
## # A tibble: 2 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1  2013     1     9      641            900      1301     1242
## 2  2013     6    15     1432           1935      1137     1607
## # ... with 12 more variables: sched_arr_time <int>, arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>
```

```
# El vuelo con menor retraso fue
arrange(flights, dep_delay)[1:2,]
```

```
## # A tibble: 2 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time
```

```
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1  2013    12     7     2040           2123       -43       40
## 2  2013     2     3     2022           2055       -33     2240
## # ... with 12 more variables: sched_arr_time <int>, arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>
```

3.2.3 Cuestión 9.3.

Ordena los vuelos de `flights` para encontrar los vuelos más rápidos. Usa el concepto de rapidez que consideres.

3.2.3.1 Solución

Por ejemplo distancia/tiempo volando

```
arrange(flights, desc(distance/air_time))
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     5    25     1709           1700         9     1923
##  2  2013     7     2     1558           1513        45     1745
##  3  2013     5    13     2040           2025        15     2225
##  4  2013     3    23     1914           1910         4     2045
##  5  2013     1    12     1559           1600        -1     1849
##  6  2013    11    17      650            655        -5     1059
##  7  2013     2    21     2355           2358        -3      412
##  8  2013    11    17      759            800        -1     1212
##  9  2013    11    16     2003           1925        38       17
## 10  2013    11    16     2349           2359       -10      402
## # ... with 336,766 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

3.2.4 Cuestión 9.4.

¿Qué vuelos tienen los trayectos más largos? Busca en Wikipedia qué dos aeropuertos del dataset alojan los vuelos más largos.

3.2.4.1 Solución

Wikipedia: Longest flights

Vuelos entre el JFK de Nueva York y el HNL, aeropuerto internacional de Honolulu en Hawaii (claro que todos los vuelos parece ser de territorio de EEUU)

3.2.5 Cuestión 9.5.

¿Qué vuelos tienen los trayectos más cortos? Busca en Wikipedia qué dos aeropuertos del dataset alojan los vuelos más largos.

3.2.5.1 Solución

Vuelos entre el EWR, Aeropuerto Internacional Libertad de Newark y LGA, Aeropuerto de La Guardia, ambos situados en el estado de Nueva York.

3.2.6 Cuestión 9.6.

Dale al coco para pensar cuantas más maneras posibles de seleccionar los campos `dep_time`, `dep_delay`, `arr_time` y `arr_delay` del dataset de `flights`.

3.2.6.1 Solución

Puedes hacerlo directamente o bien usando las diferentes funciones que hemos visto en la sección. Algunos ejemplos son:

```
select(flights,dep_time, dep_delay, arr_time, arr_delay)
```

```
## # A tibble: 336,776 x 4
##    dep_time dep_delay arr_time arr_delay
##       <int>     <dbl>    <int>     <dbl>
##  1      517         2      830        11
##  2      533         4      850        20
##  3      542         2      923        33
##  4      544        -1     1004       -18
##  5      554        -6      812       -25
##  6      554        -4      740        12
##  7      555        -5      913        19
##  8      557        -3      709       -14
##  9      557        -3      838        -8
## 10      558        -2      753         8
## # ... with 336,766 more rows
```

```
select(flights,starts_with("dep"), starts_with("arr"))
```

```
## # A tibble: 336,776 x 4
##    dep_time dep_delay arr_time arr_delay
##       <int>     <dbl>    <int>     <dbl>
##  1      517         2      830        11
##  2      533         4      850        20
##  3      542         2      923        33
##  4      544        -1     1004       -18
##  5      554        -6      812       -25
##  6      554        -4      740        12
##  7      555        -5      913        19
##  8      557        -3      709       -14
##  9      557        -3      838        -8
## 10      558        -2      753         8
## # ... with 336,766 more rows
```

```
select(flights,ends_with("time"), ends_with("delay") -starts_with("sched"),-starts_with("air") )
```

```
## # A tibble: 336,776 x 5
##    dep_time sched_dep_time arr_time sched_arr_time  year
##       <int>          <int>    <int>          <int> <int>
##  1      517            515      830            819  2013
##  2      533            529      850            830  2013
##  3      542            540      923            850  2013
##  4      544            545     1004           1022  2013
##  5      554            600      812            837  2013
```

```
##  6      554            558      740            728  2013
##  7      555            600      913            854  2013
##  8      557            600      709            723  2013
##  9      557            600      838            846  2013
## 10      558            600      753            745  2013
## # ... with 336,766 more rows
```

3.2.7 Cuestión 9.7.

¿Qué ocurre si pones el nombre de una misma variable varias veces en una `select()`?

3.2.7.1 Solución

Solo sale una vez

```
select(flights, distance, distance, distance)
```

```
## # A tibble: 336,776 x 1
##    distance
##       <dbl>
##  1     1400
##  2     1416
##  3     1089
##  4     1576
##  5      762
##  6      719
##  7     1065
##  8      229
##  9      944
## 10      733
## # ... with 336,766 more rows
```

Si la pones y la quitas pasa esto

```
select(flights, distance, distance, -distance)
```

```
## # A tibble: 336,776 x 0
```

3.2.8 Cuestión 9.8.

Investiga el uso de la función `one_of()` de dplyr.

3.2.8.1 Solución

Permite añadir las variables en string dentro de un vector. Muy útil si es el resultado de un programa que ha devuelto un array de variables que queremos seleccionar automáticamente. En el help sale el package `tydeselect`.

Por ejemplo

```
col=c("distance", "distance", "distance","delay","air_time")
flights %>% select(one_of(col)) %>% head
```

```
## Warning: Unknown columns: `delay`
```

```
## # A tibble: 6 x 2
##   distance air_time
##      <dbl>    <dbl>
## 1     1400      227
```

```
## 2     1416      227
## 3     1089      160
## 4     1576      183
## 5      762      116
## 6      719      150
```

3.2.9 Cuestión 9.9.

Investiga cómo puede ser útil la función `one_of()` de la pregunta anterior en conjunción con el vector de variables

```
c("year", "month", "day", "dep_delay", "arr_delay")
```

3.2.9.1 Solución

Pues lo probamos:

```
select(flights, one_of(c("year", "month", "day", "dep_delay", "arr_delay")))
```

```
## # A tibble: 336,776 x 5
##     year month   day dep_delay arr_delay
##    <int> <int> <int>     <dbl>     <dbl>
##  1  2013     1     1         2        11
##  2  2013     1     1         4        20
##  3  2013     1     1         2        33
##  4  2013     1     1        -1       -18
##  5  2013     1     1        -6       -25
##  6  2013     1     1        -4        12
##  7  2013     1     1        -5        19
##  8  2013     1     1        -3       -14
##  9  2013     1     1        -3        -8
## 10  2013     1     1        -2         8
## # ... with 336,766 more rows
```

Y es claro lo que hace.

3.2.10 Cuestión 9.10.

Intenta averiguar el resultado del siguiente código. Luego ejecútalo y a ver si el resultado te sorprende.

```
select(flights, contains("time"))
```

Intenta averiguar cómo lo hacen las funciones de ayuda de la `select` para tratar el caso por defecto y cómo lo puedes cambiar.

3.2.10.1 Solución

Nos devuelve todas las variables que tienen la palabra "time" en su nombre (no las que sean de tipo tiempo). Muy útil cuando queremos localizar todo lo que tiene que ver con tiempo.

```
select(flights, contains("time"))
```

```
## # A tibble: 336,776 x 6
##    dep_time sched_dep_time arr_time sched_arr_time air_time
##       <int>          <int>    <int>          <int>    <dbl>
##  1      517            515      830            819      227
##  2      533            529      850            830      227
##  3      542            540      923            850      160
##  4      544            545     1004           1022      183
```

```
##  5      554            600      812            837      116
##  6      554            558      740            728      150
##  7      555            600      913            854      158
##  8      557            600      709            723       53
##  9      557            600      838            846      140
## 10      558            600      753            745      138
## # ... with 336,766 more rows, and 1 more variable: time_hour <dttm>
```

3.3 Taller 10: Calculando nuevas variables con dplyr. Lecciones 43 y 44

Repasa los vídeos sobre las funciones `mutate`, `transmutate` y todas las variantes que se pueden usar antes de hacer los siguientes ejercicios. Preguntas de esta tarea

3.3.1 Cuestión 10.1.

El dataset de vuelos tiene dos variables, `dep_time` y `sched_dep_time` muy útiles pero difíciles de usar por cómo vienen dadas al no ser variables continuas. Fíjate que cuando pone 559, se refiere a que el vuelo salió a las 5:59...

Convierte este dato en otro más útil que represente el número de minutos desde las 00:00 horas de la media noche.

3.3.1.1 Solución

```
transmute(flights,
          dep_time, sched_dep_time,
          new_dep_time = 60*dep_time %/% 100 + dep_time %% 100,
          new_sched_dep_time = 60*sched_dep_time %/% 100 + sched_dep_time %% 100
          )
```

```
## # A tibble: 336,776 x 4
##    dep_time sched_dep_time new_dep_time new_sched_dep_time
##       <int>          <int>        <dbl>              <dbl>
##  1      517            515          317                315
##  2      533            529          333                329
##  3      542            540          342                340
##  4      544            545          344                345
##  5      554            600          354                360
##  6      554            558          354                358
##  7      555            600          355                360
##  8      557            600          357                360
##  9      557            600          357                360
## 10      558            600          358                360
## # ... with 336,766 more rows
```

3.3.2 Cuestión 10.2.

Compara las variables `air_time` contra `arr_time - dep_time`.

- ¿Qué esperas ver?
- ¿Qué ves realmente?
- ¿Se te ocurre algo para mejorarlo y corregirlo?

3.3.2.1 Solución

Pues parece que las previsiones no se cumplen ¿será por la manera de codificar el tiempo en horas minutos?, aunque no se desvía demasiado. En teoría si los datos son en minutos `air_time` y `new_air_time` deberían ser iguales ....?¿

```
transmute(flights,air_time, air_time_minutes=60*air_time %/% 100 + air_time %% 100, new_dep_time = 60*d
          new_arr_time = 60*arr_time %/% 100 + arr_time %% 100,
          new_air_time = new_arr_time - new_dep_time
          )
```

```
## # A tibble: 336,776 x 5
##    air_time air_time_minutes new_dep_time new_arr_time new_air_time
##       <dbl>            <dbl>        <dbl>        <dbl>        <dbl>
##  1      227              147          317          510          193
##  2      227              147          333          530          197
##  3      160              120          342          563          221
##  4      183              143          344          604          260
##  5      116               76          354          492          138
##  6      150              110          354          460          106
##  7      158              118          355          553          198
##  8       53               53          357          429           72
##  9      140              100          357          518          161
## 10      138               98          358          473          115
## # ... with 336,766 more rows
```

```
transmute(flights, air_time,
  air_time_minutes = 60 * air_time %/% 100 + air_time %% 100,
  sched_new_dep_time = 60 * sched_dep_time %/% 100 + sched_dep_time  %% 100,
  sched_new_arr_time = 60 * sched_arr_time %/% 100 + sched_arr_time  %% 100,
  new_air_time = sched_new_arr_time - sched_new_dep_time
)
```

```
## # A tibble: 336,776 x 5
##    air_time air_time_minutes sched_new_dep_t~ sched_new_arr_t~ new_air_time
##       <dbl>            <dbl>            <dbl>            <dbl>        <dbl>
##  1      227              147              315              499          184
##  2      227              147              329              510          181
##  3      160              120              340              530          190
##  4      183              143              345              622          277
##  5      116               76              360              517          157
##  6      150              110              358              448           90
##  7      158              118              360              534          174
##  8       53               53              360              443           83
##  9      140              100              360              526          166
## 10      138               98              360              465          105
## # ... with 336,766 more rows
```

3.3.3 Cuestión 10.3.

Compara los valores de `dep_time`, `sched_dep_time` y `dep_delay`. Cómo deberían relacionarse estos tres números? Compruébalo y haz las correcciones numéricas que necesitas.

3.3.3.1 Solución

En este caso los primero valores de `new_delayy` `dep_delay` sí parecen coincidir

```
transmute(flights,new_dep_time = 60*dep_time %/% 100 + dep_time %% 100,
          new_sched_dep_time = 60*sched_dep_time %/% 100 + sched_dep_time %% 100,
```

```
          new_delay = new_dep_time - new_sched_dep_time,
          dep_delay,new_delay==dep_delay
          )
```

```
## # A tibble: 336,776 x 5
##    new_dep_time new_sched_dep_time new_delay dep_delay `new_delay == dep_d~
##           <dbl>              <dbl>     <dbl>     <dbl> <lgl>
##  1          317                315         2         2 TRUE
##  2          333                329         4         4 TRUE
##  3          342                340         2         2 TRUE
##  4          344                345        -1        -1 TRUE
##  5          354                360        -6        -6 TRUE
##  6          354                358        -4        -4 TRUE
##  7          355                360        -5        -5 TRUE
##  8          357                360        -3        -3 TRUE
##  9          357                360        -3        -3 TRUE
## 10          358                360        -2        -2 TRUE
## # ... with 336,766 more rows
```

3.3.4 Cuestión 10.4.

Usa una de las funciones de ranking para quedarte con los 10 vuelos más retrasados de todos.

3.3.4.1 Solución

Fijaros bien: hacemos un mutate para obtener la variable `r-delay`y luego ordenamos `flights` por esa variable con `arrange`

```
arrange(mutate(flights,
               r_delay = min_rank(dep_delay)),
        r_delay
        )[1:10,]
```

```
## # A tibble: 10 x 20
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013    12     7     2040           2123       -43       40
##  2  2013     2     3     2022           2055       -33     2240
##  3  2013    11    10     1408           1440       -32     1549
##  4  2013     1    11     1900           1930       -30     2233
##  5  2013     1    29     1703           1730       -27     1947
##  6  2013     8     9      729            755       -26     1002
##  7  2013    10    23     1907           1932       -25     2143
##  8  2013     3    30     2030           2055       -25     2213
##  9  2013     3     2     1431           1455       -24     1601
## 10  2013     5     5      934            958       -24     1225
## # ... with 13 more variables: sched_arr_time <int>, arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>, r_delay <int>
```

3.3.5 Cuestión 10.5.

Aunque la ejecución te de una advertencia, qué resultado te da la operación

```
1:6 + 1:20
```

3.3.5.1 Solución

```
aux1=1:6 + 1:20
```

```
## Warning in 1:6 + 1:20: longitud de objeto mayor no es múltiplo de la
## longitud de uno menor
```

```
aux1
```

```
##  [1]  2  4  6  8 10 12  8 10 12 14 16 18 14 16 18 20 22 24 20 22
```

Es equivalente a

```
c(1:6,1:6,1:6,1,2)
```

```
##  [1] 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2
```

```
1:20
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
aux2=c(1:6,1:6,1:6,1,2) + 1:20
aux2
```

```
##  [1]  2  4  6  8 10 12  8 10 12 14 16 18 14 16 18 20 22 24 20 22
```

```
aux2==aux1
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [15] TRUE TRUE TRUE TRUE TRUE TRUE
```

```
all(aux2==aux1)
```

```
## [1] TRUE
```

3.3.6 Cuestión 10.6.

Además de todas las funciones que hemos dicho, las trigonométricas también son funciones vectoriales que podemos usar para hacer transformaciones con mutate. Investiga cuáles trae R y cuál es la sintaxis de cada una de ellas.

3.3.7 Solución

Esta información se encuentra en `help(cos)` son el seno, coseno y la tangente y sus vertientes hiperbólicas.

3.4 Taller 11. Evalución 1: Filtrado y manipulación de datos de la Sección 5, lecciones 35 a 52.

Es un taller con cuestiones globales de esta sección.

Preguntas de esta tarea

3.4.1 Cuestión 11.1.

Intenta describir con frases comprensibles el conjunto de vuelos retrasados. Intenta dar afirmaciones como por ejemplo:

- Un vuelo tiende a salir unos 20 minutos antes el 50% de las veces y a salir tarde el 50% de las veces restantes.
- Los vuelos de la compañía XX llegan siempre 20 minutos tarde.
- El 95% de los vuelos a HNL llegan a tiempo, pero el 5% restante se retrasan más de 3 horas.

Intenta dar por lo menos 5 afirmaciones verídicas en base a los datos que tenemos disponibles.

3.4.1.1 Solución

```
# Un vuelo tiende a salir unos 20 minutos antes el 50% de las veces y a salir tarde el 50% de las veces
flights %>% summarise(median = median(dep_delay,na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##   median
##    <dbl>
## 1     -2
```

```
# Los vuelos de la compañía XX llegan siempre 20 minutos tarde.
flights %>% transmute(carrier,delay_gt_20=dep_delay>20,delay_lt_20=dep_delay<=20)%>%
  group_by(carrier) %>% summarise(n_delay_gt=sum(delay_gt_20,na.rm=TRUE),n_delay_lt=sum(delay_lt_20,na.
```

```
## # A tibble: 16 x 4
##    carrier n_delay_gt n_delay_lt percent_gt_20
##    <chr>        <int>      <int>         <dbl>
##  1 EV           14148      37208         27.6
##  2 YV             144        401         26.4
##  3 F9             163        519         23.9
##  4 9E            4018      13398         23.1
##  5 WN            2740       9343         22.7
##  6 FL             716       2471         22.5
##  7 OO               6         23         20.7
##  8 B6           10728      43441         19.8
##  9 MQ            4720      20443         18.8
## 10 UA           10236      47743         17.6
## 11 VX             753       4378         14.7
## 12 AA            4443      27650         13.8
## 13 DL            6611      41150         13.8
## 14 AS              78        634         11.0
## 15 US            2108      17765         10.6
## 16 HA              21        321          6.14
```

```
# El 95% de los vuelos a HNL (tomaremos HA pues NHL no aparece) llegan a tiempo, pero el 5% restante se

flights %>% filter(dest=="HNL") %>%
  transmute(dest,no_delay=dep_delay<=0,delay_gt_180=dep_delay>=180)%>%
  summarise(
    no_delay_percent=sum(no_delay,na.rm=TRUE)/(sum(no_delay,na.rm=TRUE)+sum(!no_delay,na.rm=TRUE)),   d
    )
```

```
## # A tibble: 1 x 2
##   no_delay_percent delay_gt_180
##              <dbl>        <dbl>
## 1            0.599       0.0113
```

3.4.2 Cuestión 11.2.

Da una versión equivalente a las pipes siguientes sin usar la función count:

```
not_cancelled <- flights %>%  count(dest)
not_cancelled <- count(tailnum, wt = distance)
```

3.4.2.1 Solución

Primero agrupamos con `group_by()` y luego contamos con `tally` en el primer caso sin pesos y en el segundo con pesos la variable `distance` (`help(count)`) sugiere estas instrucciones como definición del atajo `count()`)

```
not_cancelled <-flights %>% group_by(dest) %>% tally() #
not_cancelled_enunciado <- flights %>%  count(dest)
not_cancelled
```

```
## # A tibble: 105 x 2
##    dest      n
##    <chr> <int>
##  1 ABQ     254
##  2 ACK     265
##  3 ALB     439
##  4 ANC       8
##  5 ATL   17215
##  6 AUS    2439
##  7 AVL     275
##  8 BDL     443
##  9 BGR     375
## 10 BHM     297
## # ... with 95 more rows
```

```
not_cancelled_enunciado
```

```
## # A tibble: 105 x 2
##    dest      n
##    <chr> <int>
##  1 ABQ     254
##  2 ACK     265
##  3 ALB     439
##  4 ANC       8
##  5 ATL   17215
##  6 AUS    2439
##  7 AVL     275
##  8 BDL     443
##  9 BGR     375
## 10 BHM     297
## # ... with 95 more rows
```

```
all(not_cancelled==not_cancelled_enunciado)
```

```
## [1] TRUE
```

Para la segunda puede valer este código

```
not_cancelled_enuciado <- flights %>% count(tailnum, wt = distance) #https://en.wikipedia.org/wiki/Tail
not_cancelled <- flights %>%   group_by(tailnum) %>% tally(wt=distance)
not_cancelled
```

```
## # A tibble: 4,044 x 2
##    tailnum       n
##    <chr>     <dbl>
##  1 <NA>    1784167
##  2 D942DN     3418
##  3 NOEGMQ   250866
##  4 N10156   115966
##  5 N102UW    25722
##  6 N103US    24619
##  7 N104UW    25157
##  8 N10575   150194
```

```
##  9 N105UW    23618
## 10 N107US    21677
## # ... with 4,034 more rows
```

```
not_cancelled_enuciado
```

```
## # A tibble: 4,044 x 2
##    tailnum       n
##    <chr>     <dbl>
##  1 <NA>    1784167
##  2 D942DN     3418
##  3 N0EGMQ   250866
##  4 N10156   115966
##  5 N102UW    25722
##  6 N103US    24619
##  7 N104UW    25157
##  8 N10575   150194
##  9 N105UW    23618
## 10 N107US    21677
## # ... with 4,034 more rows
```

3.4.3 Cuestión 11.3.

Para definir un vuelo cancelado hemos usado la función

```
(is.na(dep_delay) | is.na(arr_delay))
```

Intenta dar una definición que sea mejor, ya que la nuestra es un poco subóptima. ¿Cuál es la columna más importante?

3.4.3.1 Solución

Pues otra vez son conjeturas. Veamos que variables con la cadena `timesen` su nombre tenemos (ver un ejercicio anterior)

```
select(flights, contains("time"))
```

```
## # A tibble: 336,776 x 6
##    dep_time sched_dep_time arr_time sched_arr_time air_time
##       <int>          <int>    <int>          <int>    <dbl>
##  1      517            515      830            819      227
##  2      533            529      850            830      227
##  3      542            540      923            850      160
##  4      544            545     1004           1022      183
##  5      554            600      812            837      116
##  6      554            558      740            728      150
##  7      555            600      913            854      158
##  8      557            600      709            723       53
##  9      557            600      838            846      140
## 10      558            600      753            745      138
## # ... with 336,766 more rows, and 1 more variable: time_hour <dttm>
```

Ahora podemos definir qué vuelos consideramos cancelados según qué variables son NA. Antes comprobemos que los NA no son coincidentes

```
summary_NA<- function(v,name_v) {
  tibble(
    column     = name_v,
```

```
    na_num     = sum(is.na(v))
  )}
not_cancelled <- select(flights, contains("time")) %>% imap_dfr(summary_NA)
not_cancelled
```

```
## # A tibble: 6 x 2
##   column         na_num
##   <chr>           <int>
## 1 dep_time         8255
## 2 sched_dep_time      0
## 3 arr_time         8713
## 4 sched_arr_time      0
## 5 air_time         9430
## 6 time_hour           0
```

3.4.4 Cuestión 11.4.

Investiga si existe algún patrón del número de vuelos que se cancelan cada día.

Investiga si la proporción de vuelos cancelados está relacionada con el retraso promedio por día en los vuelos.

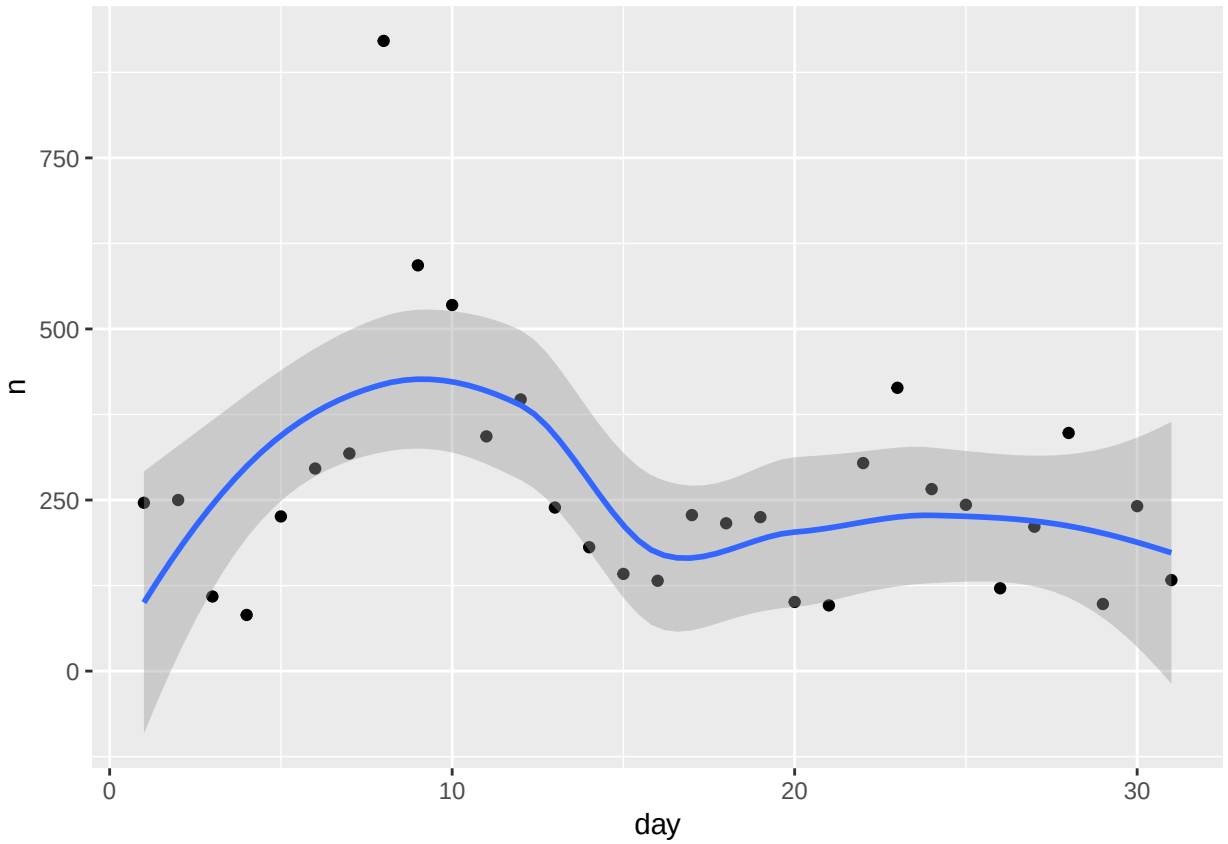
Investiga si la proporción de vuelos cancelados está relacionada con el retraso promedio por aeropuerto en los vuelos.

¿Qué compañía aérea sufre los peores retrasos?

3.4.4.1 Solución

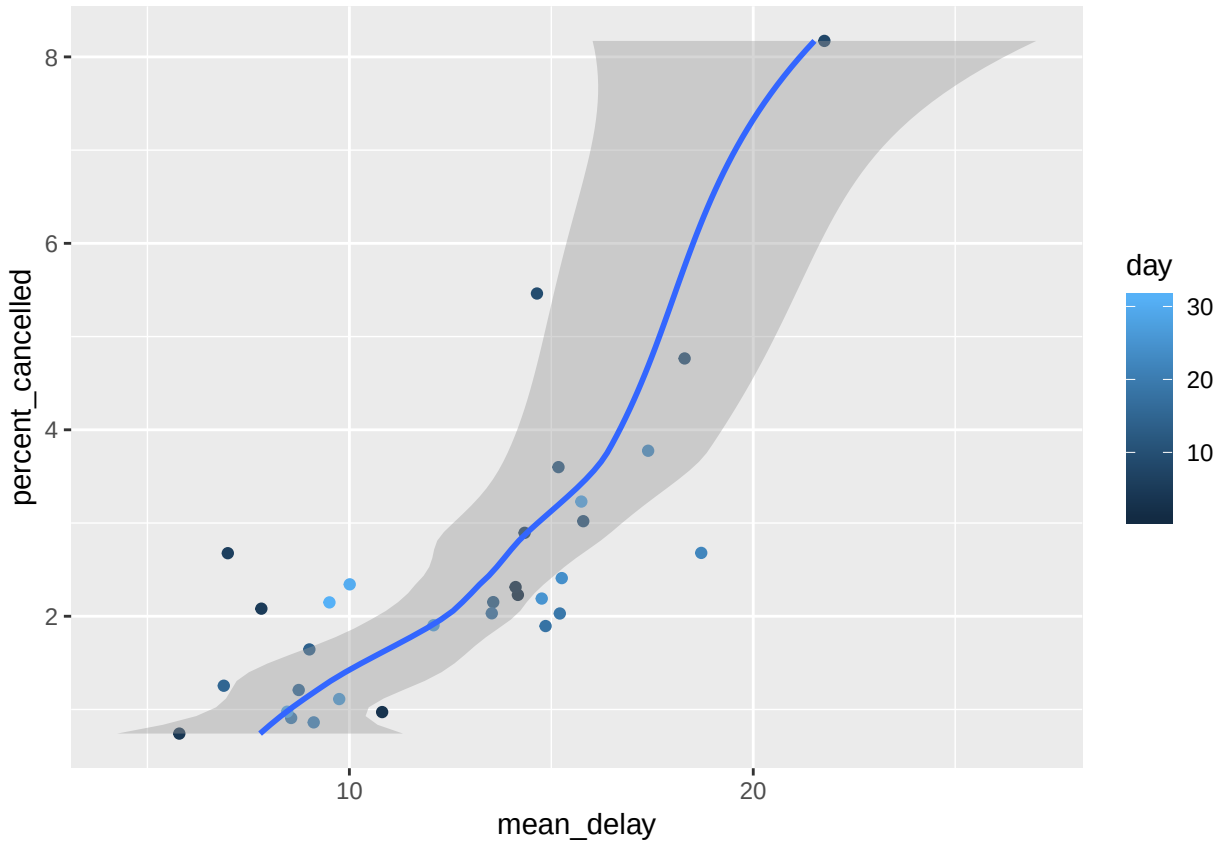
```
filter(flights,is.na(dep_time)) %>% group_by(day) %>% count() %>% ggplot(aes(x=day,y=n))+geom_point()+g
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
# Investiga si la proporción de vuelos cancelados está relacionada con el retraso promedio por día en l
mutate(flights ,cancelled=is.na(dep_time),not_cancelled=!cancelled) %>%  group_by(day) %>%  summarise(p
aux %>% ggplot(aes(x=percent_cancelled,y=mean_delay,color=day))+geom_point(shape=19)+geom_smooth()+coor
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
summary(lm(aux$mean_delay~aux$percent_cancelled))
```

```
##
## Call:
## lm(formula = aux$mean_delay ~ aux$percent_cancelled)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.0717 -1.9737  0.5104  1.8630  5.6443
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)             7.9076     0.9286   8.516 2.21e-09 ***
## aux$percent_cancelled   1.9258     0.3236   5.951 1.82e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.717 on 29 degrees of freedom
## Multiple R-squared:  0.5498, Adjusted R-squared:  0.5342
## F-statistic: 35.41 on 1 and 29 DF,  p-value: 1.823e-06
```

3.4.5 Cuestión 11.5.

Difícil: Intenta desentrañar los efectos que producen los retrasos por culpa de malos aeropuertos vs malas compañías aéreas. Por ejemplo, intenta usar

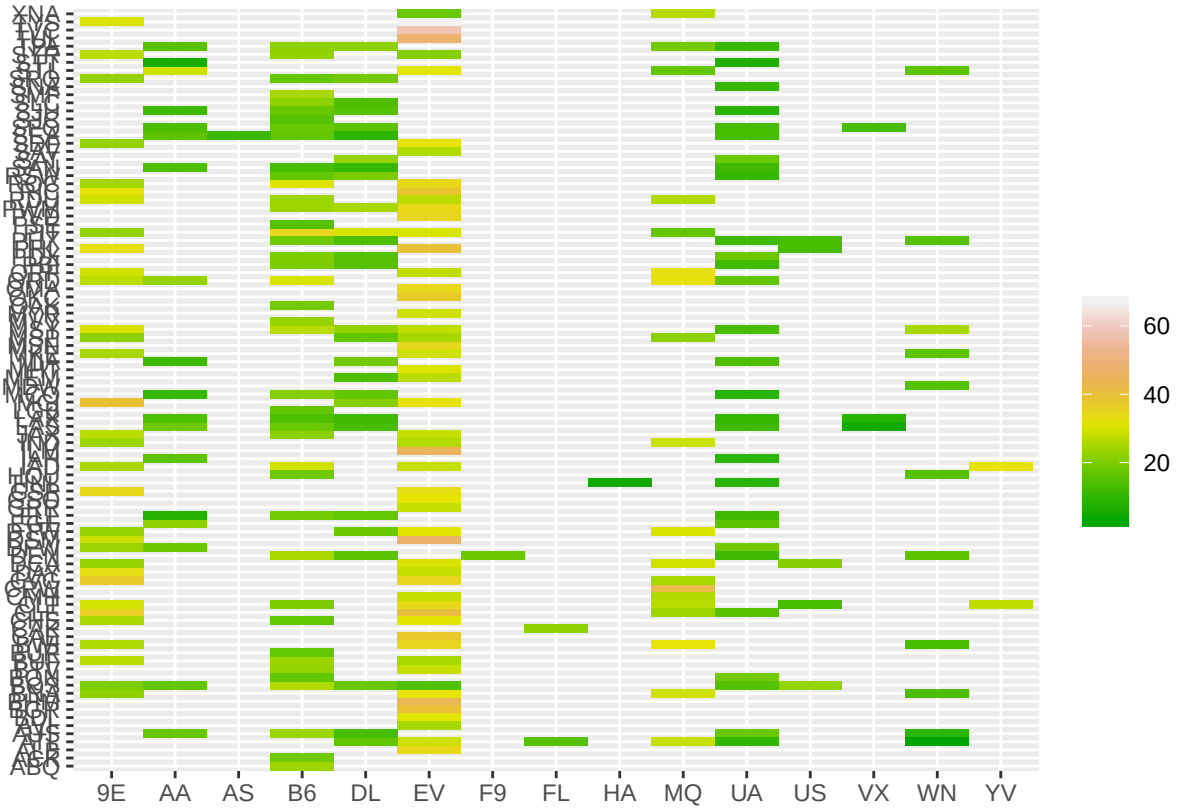
```
flights %>% group_by(carrier, dest) %>% summarise(n())
```

```
## # A tibble: 314 x 3
## # Groups:   carrier [16]
##    carrier dest  `n()`
##    <chr>   <chr> <int>
##  1 9E      ATL      59
##  2 9E      AUS       2
##  3 9E      AVL      10
##  4 9E      BGR       1
##  5 9E      BNA     474
##  6 9E      BOS     914
##  7 9E      BTV       2
##  8 9E      BUF     833
##  9 9E      BWI     856
## 10 9E      CAE       3
## # ... with 304 more rows
```

3.4.5.1 Solución

```
flights %>% filter(!is.na(dep_delay)&dep_delay>0)%>%group_by(carrier, dest) %>% summarise(abs_freq=n(),m
```

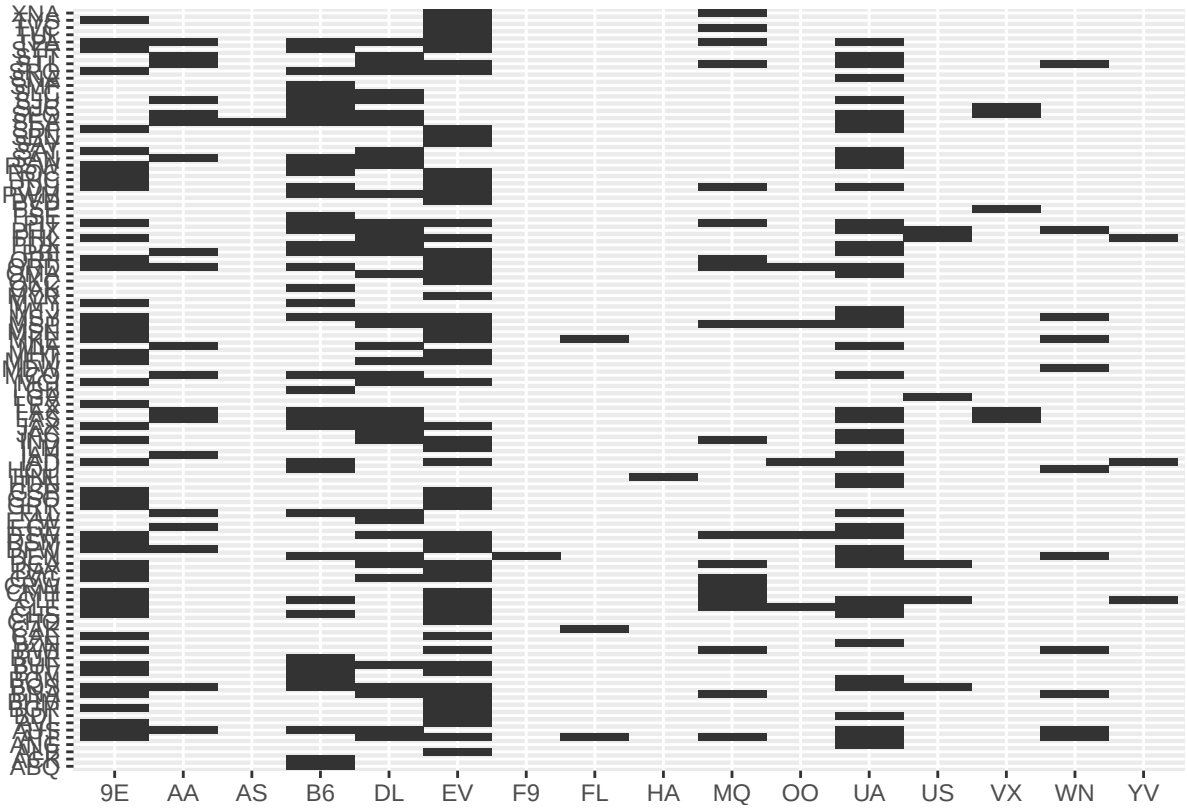
```
ggplot(aux, aes(x = carrier, y = dest,fill=median)) +
  geom_tile() +
  scale_fill_gradientn(name = "", colors = terrain.colors(10)) +
  scale_x_discrete(name = "") +
  scale_y_discrete(name = "")
```



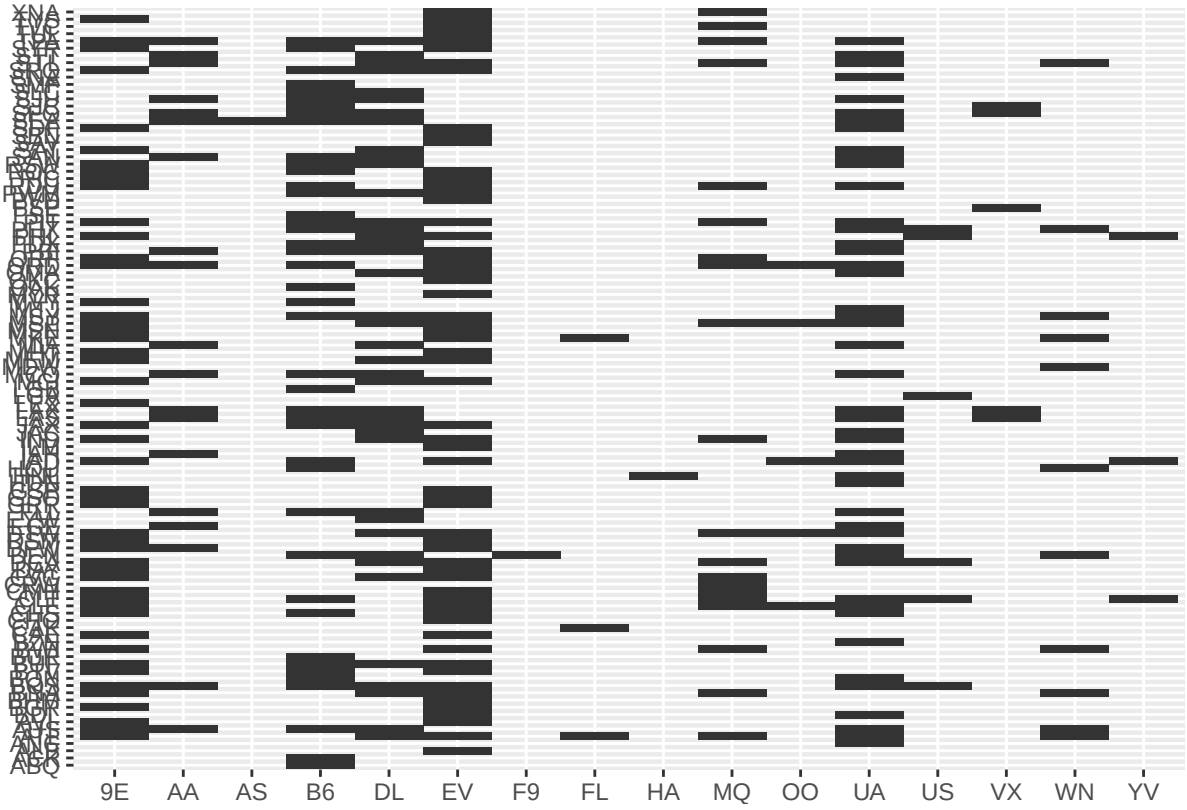

```
flights %>% group_by(carrier, dest) %>% summarise(abs_freq=n())%>% arrange(desc(abs_freq))-> aux
aux
```

```
## # A tibble: 314 x 3
## # Groups:   carrier [16]
##    carrier dest  abs_freq
##    <chr>   <chr>    <int>
##  1 DL      ATL      10571
##  2 US      CLT       8632
##  3 AA      DFW       7257
##  4 AA      MIA       7234
##  5 UA      ORD       6984
##  6 UA      IAH       6924
##  7 UA      SFO       6819
##  8 B6      FLL       6563
##  9 B6      MCO       6472
## 10 AA      ORD       6059
## # ... with 304 more rows
```

```
ggplot(aux, aes(x = carrier, y = dest),fill=abs_freq) +
  geom_tile() +
  scale_fill_gradientn(name = "", colors = terrain.colors(10)) +
  scale_x_discrete(name = "") +
  scale_y_discrete(name = "")
```




```
ggplot(aux, aes(x = carrier, y = dest)) +
  geom_tile() +
  scale_fill_gradientn(name = "", colors = terrain.colors(10)) +
  scale_x_discrete(name = "") +
  scale_y_discrete(name = "")
```



3.4.6 Cuestión 11.6.

¿Qué hace el parámetro sort como argumento de `count()`? ¿Cuando puede sernos útil?

Vuelve a la lista de funciones útiles para filtrar y mutar y describe cómo cada operación cambia cuando la juntamos con un `group_by`.

3.4.6.1 Solución

```
ggplot(aux, aes(x=carrier, y=dest, group=carrier)) +
geom_col("abs_freq", position='dodge')
```

3.4.7 Cuestión 11.7.

Vamos a por los peores aviones. Investiga el top 10 de qué aviones (número de cola y compañía) llegaron más tarde a su destino.

3.4.7.1 Solución

```
flights %>% transmute(tailnum,carrier,arr_delay)%>% arrange(desc(arr_delay))%>% slice(1:10)
```

```
## # A tibble: 10 x 3
##    tailnum carrier arr_delay
##    <chr>   <chr>       <dbl>
##  1 N384HA  HA           1272
##  2 N504MQ  MQ           1127
##  3 N517MQ  MQ           1109
##  4 N338AA  AA           1007
##  5 N665MQ  MQ            989
##  6 N959DL  DL            931
```

```
##  7 N927DA  DL            915
##  8 N6716C  DL            895
##  9 N5DMAA  AA            878
## 10 N523MQ  MQ            875
```

3.4.8 Cuestión 11.8.

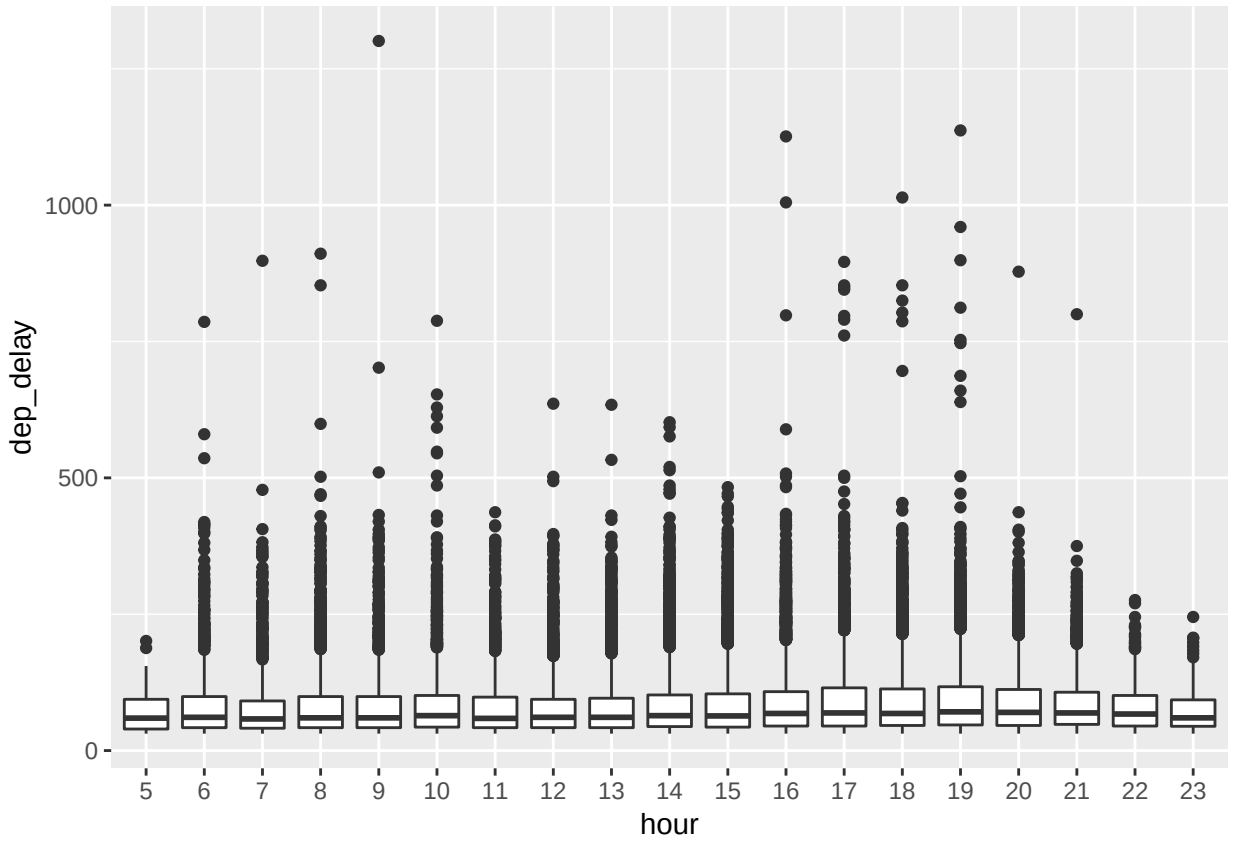
Queremos saber qué hora del día nos conviene volar si queremos evitar los retrasos en la salida.

Difícil: Queremos saber qué día de la semana nos conviene volar si queremos evitar los retrasos en la salida.

3.4.8.1 Solución

Atención solo cogemos los retrasos positivos.

```
# horas del día con menos retraso: he decidido filtrar por un umbral y fdibujar los diagramas de caja
umbral=30# 30 minutos
flights %>%  transmute(hour=as.factor(hour),dep_delay) %>% filter(!is.na(dep_delay)&dep_delay>umbral) %
```



También podemos calcular la media de los retrasos positivos por hora

```
flights %>% transmute(hour,dep_delay) %>% filter(!is.na(dep_delay) & dep_delay>0) %>% group_by(hour) %>%
  summarise(mean=mean(dep_delay),median=median(dep_delay),standar_dev=sd(dep_delay),IQR=IQR(dep_delay),m
positive_dep_delay_by_hour
```

```
## # A tibble: 19 x 9
##     hour  mean median standar_dev   IQR   min   max Q0.25 Q0.75
##    <dbl> <dbl>  <dbl>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1     5  15.3     6          26.4    12     1   201     3    15
##  2     6  24.2     9          44.0    21     1   786     3    24
```

```
##  3     7  24.1    9          42.8    23     1   898     3    26
##  4     8  29.9   13          49.7    29     1   911     4    33
##  5     9  29.7   13          50.2    30     1  1301     4    34
##  6    10  32.6   13          52.6    33     1   788     5    38
##  7    11  32.5   15          48.0    34     1   437     5    39
##  8    12  32.3   16          46.8    35     1   636     5    40
##  9    13  33.5   16          47.4    34     1   634     6    40
## 10    14  37.1   17          53.0    40     1   602     6    46
## 11    15  38.8   19          52.8    42     1   483     7    49
## 12    16  43.4   22          58.1    49     1  1126     8    57
## 13    17  45.3   23          60.6    51     1   896     8    59
## 14    18  46.5   25          59.5    53     1  1014     8    61
## 15    19  51.1   29          61.8    60     1  1137    10    70
## 16    20  49.6   30          54.8    58     1   878    11    69
## 17    21  50.3   34          51.3    60     1   800    12    72
## 18    22  46.5   31.5        46.1    56     1   276    12    68
## 19    23  38.0   22          42.1    44     1   245     8    52
```

Hay que utilizar el package lubridate para extraer el día de la semana

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
flights$time_hour[1:2]
```

```
## [1] "2013-01-01 05:00:00 EST" "2013-01-01 05:00:00 EST"
```

```
wday(flights$time_hour[1:2])
```

```
## [1] 3 3
```

```
Sys.Date()
```

```
## [1] "2019-10-09"
```

```
wday(Sys.Date())
```

```
## [1] 4
```

```
wday(Sys.Date(),week_start = getOption("lubridate.week.start", 1))
```

```
## [1] 3
```

```
wday(Sys.Date(),label=TRUE,week_start = getOption("lubridate.week.start", 1))
```

```
## [1] mié
## Levels: lun < mar < mié < jue < vie < sáb < dom
```

```
flights %>% transmute(day=wday(time_hour,label=TRUE,week_start = getOption("lubridate.week.start", 1)),
filter(!is.na(dep_delay)&dep_delay>0) %>% group_by(day)%>%    summarise(mean=mean(dep_delay),median=medi
positive_dep_delay_by_week_day
```

```
## # A tibble: 7 x 9
##   day    mean median standar_dev   IQR   min   max Q0.25 Q0.75
##   <ord> <dbl>  <dbl>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
```

```
## 1 lun    43.5     21        57.6    50     1  1005     7    57
## 2 mar    37.4     19        49.0    44     1   853     6    50
## 3 mié    39.5     18        56.0    43     1  1301     6    49
## 4 jue    43.1     21        58.2    49     1  1126     7    56
## 5 vie    40.4     20        54.3    46     1  1014     7    53
## 6 sáb    30.5     14        45.7    31     1  1137     5    36
## 7 dom    37.5     18        54.0    40     1   911     6    46
```

3.4.9 Cuestión 11.9.

Para cada destino, calcula el total de minutos de retraso acumulado. Para cada uno de ellos, calcula la proporción del total de retraso para dicho destino.

3.4.9.1 Solución

```
flights %>% transmute(dest,dep_delay)%>% filter(dep_delay>0)%>%group_by(dest) %>%
  summarise(sum=sum(dep_delay,na.rm=TRUE))%>%
  mutate(prop_delay=sum/sum(sum))%>% arrange(sum)-> aux
```

3.4.10 Cuestión 11.10.

Los retrasos suelen estar correlacionados con el tiempo. Aunque el problema que ha causado el primer retraso de un avión se resuelva, el resto de vuelos se retrasan para que salgan primero los aviones que debían haber partido antes. Intenta usar la función `lag()` explora cómo el retraso de un avión se relaciona con el retraso del avión inmediatamente anterior o posterior.

3.4.11 Solución

Seleccionaremos las salidas de JFK y los retrasos en la salida. Necesitamos ordenar los vuelos por salida

```
flights%>% filter(origin=="JFK") %>% transmute(origin, time_hour,dep_delay) %>% arrange(time_hour)-> fli
lapply(1:10,function(x) cor(flights_time_order$dep_delay,lag(flights_time_order$dep_delay,x),use="compl
correlation_delay_lag<- unlist(correlation_delay_lag)
names(correlation_delay_lag)<-paste("lag",1:10,sep="_")
correlation_delay_lag
```

```
##      lag_1      lag_2      lag_3      lag_4      lag_5      lag_6
## 0.55227212 0.38265103 0.28005542 0.21765785 0.17309678 0.14035150
##      lag_7      lag_8      lag_9     lag_10
## 0.11570897 0.09918925 0.09461354 0.08716629
```

Vemos como las correlaciones entre los retrasos decrecen

3.4.12 Cuestión 11.11.

Vamos a por los destinos esta vez. Localiza vuelos que llegaron 'demasiado rápido' a sus destinos. Seguramente, el becario se equivocó al introducir el tiempo de vuelo y se trate de un error en los datos. Calcula para ello el cociente entre el tiempo en el aire de cada vuelo relativo al tiempo de vuelo del avión que tardó menos en llegar a dicho destino. ¿Qué vuelos fueron los que más se retrasaron en el aire?

3.4.12.1 Solución

```
flights %>% filter(origin=="JFK",!is.na(air_time))  -> JFK_fli
JFK_fli %>% right_join( JFK_fli %>% group_by(dest) %>% summarise(min_air_time=min(air_time),key="dest")
  transmute(dest,tailnum,carrier,sched_dep_time,sched_arr_time,air_time, min_air_time,rel_air_time_by_d
```

```
## Joining, by = "dest"
```

```
JFK_fli
```

```
## # A tibble: 109,079 x 8
##    dest  tailnum carrier sched_dep_time sched_arr_time air_time
##    <chr> <chr>   <chr>            <int>          <int>    <dbl>
##  1 ACK   N328JB  B6                 800            909      141
##  2 BOS   N3FKAA  AA                1605           1740       96
##  3 BOS   N346NB  DL                1200           1317       91
##  4 BOS   N913XJ  9E                 840           1003       86
##  5 BOS   N3DRAA  AA                1245           1355       86
##  6 BOS   N3FEAA  AA                1245           1350       80
##  7 BOS   N279JB  B6                1645           1813       77
##  8 BOS   N3FJAA  AA                1600           1720       76
##  9 PHL   N8932C  9E                 940           1051       61
## 10 DCA   N813MQ  MQ                 755            910       97
## # ... with 109,069 more rows, and 2 more variables: min_air_time <dbl>,
## #   rel_air_time_by_dest <dbl>
```

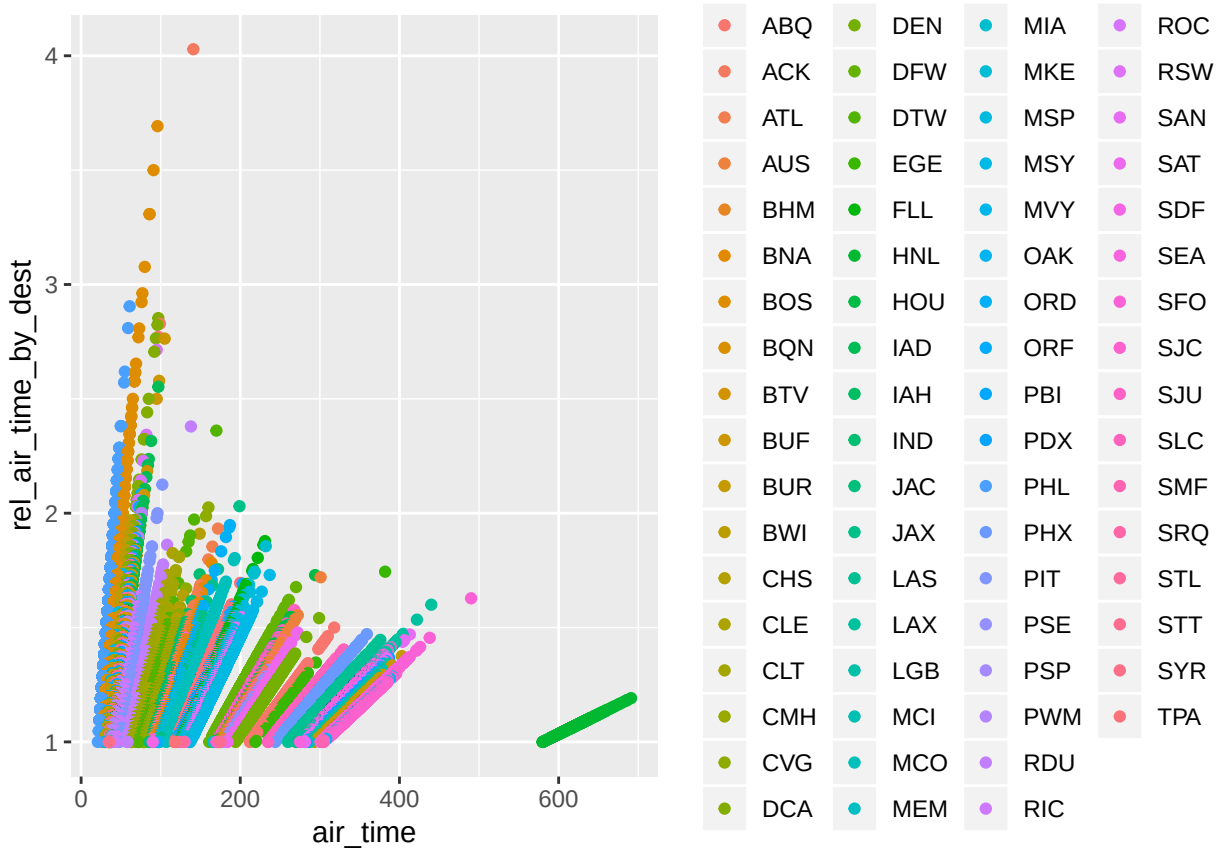
Los vuelos a los aeropuertos con más retraso relativo son de los aeropuertos

```
airports[airports$faa %in% c("ACK","BOS","DCA"),]
```

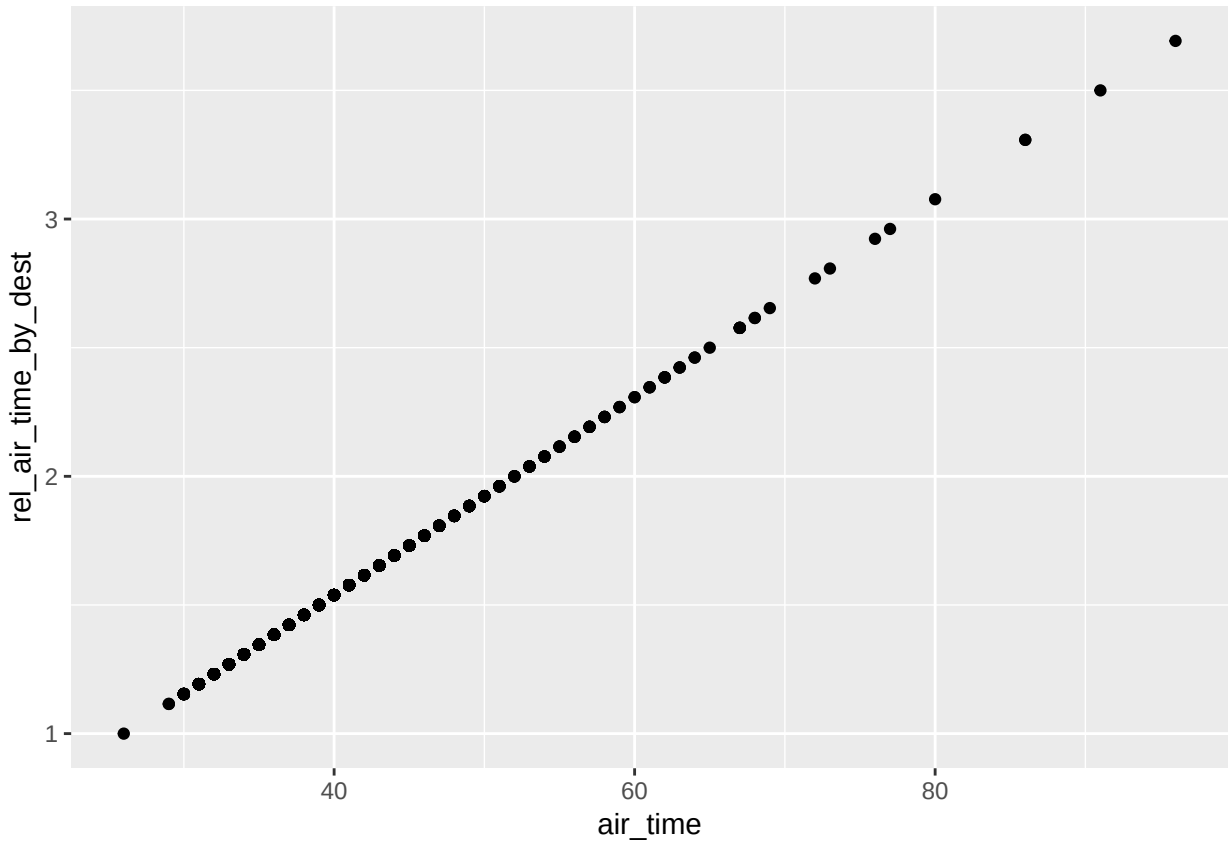
```
## # A tibble: 3 x 8
##   faa   name                       lat   lon   alt    tz dst   tzone
##   <chr> <chr>                    <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1 ACK   Nantucket Mem             41.3 -70.1    48    -5 A     America/New~
## 2 BOS   General Edward Lawrence~  42.4 -71.0    19    -5 A     America/New~
## 3 DCA   Ronald Reagan Washingto~  38.9 -77.0    15    -5 A     America/New~
```

Un gráfico de los tiempos relativos contra los absolutos (no interpretéis mucho las rectas pues lo son y=x/min), el gráfico es mejorable.

```
JFK_fli %>% ggplot(aes(x=air_time,y=rel_air_time_by_dest))+ geom_point(aes(col=dest))
```




```
JFK_fli %>% filter(dest=="BOS")%>%
 ggplot(aes(x=air_time,y=rel_air_time_by_dest))+ geom_point()
```



3.4.13 Cuestión 11.12.

Encuentra todos los destinos a los que vuelan dos o más compañías y para cada uno de ellos, crea un ranking de las mejores compañías para volar a cada destino (utiliza el criterio que consideres más conveniente como probabilidad de retraso, velocidad o tiempo de vuelo, número de vuelos al año..)

Finalmente, para cada avión (basándonos en el número de cola) cuenta el número de vuelos que hace antes de sufrir su primer retraso de más de una hora. Valora entonces la fiabilidad del avión o de la compañía aérea asociada al mismo.

3.4.13.1 Solución

```
flights %>% group_by(dest, carrier) %>% count() %>% filter(n >= 2) -> aux
filter_dest_carrier = paste(aux$dest, aux$carrier)
filter_dest_carrier
```

```
##   [1] "ABQ B6" "ACK B6" "ALB EV" "ANC UA" "ATL 9E" "ATL DL" "ATL EV"
##   [8] "ATL FL" "ATL MQ" "ATL UA" "ATL WN" "AUS 9E" "AUS AA" "AUS B6"
##  [15] "AUS DL" "AUS UA" "AUS WN" "AVL 9E" "AVL EV" "BDL EV" "BDL UA"
##  [22] "BGR EV" "BHM EV" "BNA 9E" "BNA EV" "BNA MQ" "BNA WN" "BOS 9E"
##  [29] "BOS AA" "BOS B6" "BOS DL" "BOS EV" "BOS UA" "BOS US" "BQN B6"
##  [36] "BQN UA" "BTV 9E" "BTV B6" "BTV EV" "BUF 9E" "BUF B6" "BUF DL"
##  [43] "BUF EV" "BUR B6" "BWI 9E" "BWI EV" "BWI MQ" "BWI WN" "BZN UA"
##  [50] "CAE 9E" "CAE EV" "CAK FL" "CHO EV" "CHS 9E" "CHS B6" "CHS EV"
##  [57] "CLE 9E" "CLE EV" "CLE MQ" "CLE OO" "CLE UA" "CLT 9E" "CLT B6"
##  [64] "CLT EV" "CLT MQ" "CLT UA" "CLT US" "CLT YV" "CMH 9E" "CMH EV"
##  [71] "CMH MQ" "CRW MQ" "CVG 9E" "CVG DL" "CVG EV" "CVG MQ" "DAY 9E"
##  [78] "DAY EV" "DCA 9E" "DCA DL" "DCA EV" "DCA MQ" "DCA UA" "DCA US"
```

```
##  [85] "DEN B6" "DEN DL" "DEN F9" "DEN UA" "DEN WN" "DFW 9E" "DFW AA"
##  [92] "DFW EV" "DFW UA" "DSM 9E" "DSM EV" "DTW 9E" "DTW DL" "DTW EV"
##  [99] "DTW MQ" "DTW OO" "EGE AA" "EGE UA" "EYW DL" "FLL AA" "FLL B6"
## [106] "FLL DL" "FLL UA" "GRR 9E" "GRR EV" "GSO EV" "GSP 9E" "GSP EV"
## [113] "HDN UA" "HNL HA" "HNL UA" "HOU B6" "HOU WN" "IAD 9E" "IAD B6"
## [120] "IAD EV" "IAD YV" "IAH AA" "IAH UA" "ILM EV" "IND 9E" "IND DL"
## [127] "IND EV" "IND MQ" "IND UA" "JAC DL" "JAC UA" "JAX 9E" "JAX B6"
## [134] "JAX EV" "LAS AA" "LAS B6" "LAS DL" "LAS UA" "LAS VX" "LAX AA"
## [141] "LAX B6" "LAX DL" "LAX UA" "LAX VX" "LGB B6" "MCI 9E" "MCI DL"
## [148] "MCI EV" "MCO AA" "MCO B6" "MCO DL" "MCO UA" "MDW WN" "MEM 9E"
## [155] "MEM DL" "MEM EV" "MHT 9E" "MHT EV" "MIA AA" "MIA DL" "MIA UA"
## [162] "MKE 9E" "MKE EV" "MKE FL" "MKE WN" "MSN EV" "MSP 9E" "MSP DL"
## [169] "MSP EV" "MSP MQ" "MSP OO" "MSP UA" "MSY 9E" "MSY B6" "MSY DL"
## [176] "MSY EV" "MSY UA" "MSY WN" "MTJ UA" "MVY 9E" "MVY B6" "MYR EV"
## [183] "OAK B6" "OKC EV" "OMA EV" "OMA UA" "ORD 9E" "ORD AA" "ORD B6"
## [190] "ORD EV" "ORD MQ" "ORD UA" "ORF 9E" "ORF EV" "ORF MQ" "PBI AA"
## [197] "PBI B6" "PBI DL" "PBI EV" "PBI UA" "PDX B6" "PDX DL" "PDX UA"
## [204] "PHL 9E" "PHL DL" "PHL EV" "PHL US" "PHL YV" "PHX B6" "PHX DL"
## [211] "PHX UA" "PHX US" "PHX WN" "PIT 9E" "PIT B6" "PIT DL" "PIT EV"
## [218] "PIT MQ" "PIT UA" "PSE B6" "PSP VX" "PVD EV" "PWM B6" "PWM DL"
## [225] "PWM EV" "RDU 9E" "RDU B6" "RDU EV" "RDU MQ" "RIC 9E" "RIC EV"
## [232] "ROC 9E" "ROC B6" "ROC EV" "RSW 9E" "RSW B6" "RSW DL" "RSW UA"
## [239] "SAN AA" "SAN B6" "SAN DL" "SAN UA" "SAT 9E" "SAT DL" "SAT UA"
## [246] "SAV EV" "SBN EV" "SDF 9E" "SDF EV" "SDF UA" "SEA AA" "SEA AS"
## [253] "SEA B6" "SEA DL" "SEA UA" "SFO AA" "SFO B6" "SFO DL" "SFO UA"
## [260] "SFO VX" "SJC B6" "SJU AA" "SJU B6" "SJU DL" "SJU UA" "SLC B6"
## [267] "SLC DL" "SMF B6" "SNA UA" "SRQ 9E" "SRQ B6" "SRQ DL" "SRQ EV"
## [274] "STL AA" "STL EV" "STL MQ" "STL UA" "STL WN" "STT AA" "STT DL"
## [281] "STT UA" "SYR 9E" "SYR B6" "SYR EV" "TPA 9E" "TPA AA" "TPA B6"
## [288] "TPA DL" "TPA MQ" "TPA UA" "TUL EV" "TVC EV" "TVC MQ" "TYS 9E"
## [295] "TYS EV" "XNA EV" "XNA MQ"
```

```
flights %>% filter(paste(dest, carrier) %in% filter_dest_carrier &
                     !is.na(arr_delay)) %>%
  transmute(dest, carrier, arr_delay, flights_delayeds = arr_delay >= 5) %>% group_by(dest, carrier) %>%
  summarise(
    total_flights = n(),
    total_flights_delayeds = sum(flights_delayeds),
    percent_flights_delayeds = 100 * total_flights_delayeds / total_flights
  ) %>% arrange(percent_flights_delayeds, total_flights) %>% print(pp, n =
```

```
## # A tibble: 297 x 5
## # Groups:   dest [103]
##     dest  carrier total_flights total_flights_delaye~ percent_flights_dela~
##     <chr> <chr>           <int>                <int>                  <dbl>
##   1 BTV   9E                  2                    0                      0
##   2 DCA   DL                  2                    0                      0
##   3 IND   DL                  2                    0                      0
##   4 MSP   UA                  2                    0                      0
##   5 OMA   UA                  2                    0                      0
##   6 PHL   DL                  2                    0                      0
##   7 BUF   DL                  3                    0                      0
##   8 IND   UA                  3                    0                      0
##   9 MEM   9E                  3                    0                      0
##  10 SDF   UA                  3                    0                      0
```

```
##  11 TPA   9E                3                   0                   0
##  12 MHT   9E               10                   0                   0
##  13 STT   DL               29                   1                3.45
##  14 DFW   EV                8                   1                12.5
##  15 PBI   AA               80                  11                13.8
##  16 AUS   WN              293                  42                14.3
##  17 PSP   VX               18                   3                16.7
##  18 SRQ   EV               28                   5                17.9
##  19 BOS   EV              156                  29                18.6
##  20 AVL   9E               10                   2                20
##  21 MCI   DL               80                  16                20
##  22 SRQ   DL              263                  53                20.2
##  23 SEA   DL             1202                 248                20.6
##  24 LAS   VX              364                  76                20.9
##  25 RSW   9E               67                  14                20.9
##  26 SJU   DL             1294                 274                21.2
##  27 MVY   9E               65                  14                21.5
##  28 BOS   US             4002                 879                22.0
##  29 SEA   AS              709                 159                22.4
##  30 HNL   HA              342                  78                22.8
##  31 DAY   9E              342                  79                23.1
##  32 BOS   DL              961                 222                23.1
##  33 FLL   AA              179                  42                23.5
##  34 EGE   UA              106                  25                23.6
##  35 DTW   DL             3833                 909                23.7
##  36 CLE   OO               21                   5                23.8
##  37 SFO   DL             1848                 443                24.0
##  38 SDF   9E              250                  60                24
##  39 MVY   B6              145                  35                24.1
##  40 MSY   UA              265                  64                24.2
##  41 STT   AA              301                  74                24.6
##  42 SNA   UA              812                 200                24.6
##  43 MSP   OO                4                   1                25
##  44 PHL   YV                8                   2                25
##  45 PHX   DL              467                 117                25.1
##  46 MIA   AA             7143                1798                25.2
##  47 STL   AA              864                 219                25.3
##  48 BOS   AA             1428                 362                25.4
##  49 LAS   DL             1661                 434                26.1
##  50 AUS   DL              352                  92                26.1
##  51 SLC   DL             2087                 548                26.3
##  52 GRR   9E               38                  10                26.3
##  53 SAT   9E               34                   9                26.5
##  54 MCO   UA             3191                 852                26.7
##  55 ORD   AA             5846                1562                26.7
##  56 LAX   DL             2487                 667                26.8
##  57 MEM   DL              424                 114                26.9
##  58 DCA   9E             1011                 272                26.9
##  59 SAN   DL              567                 153                27.0
##  60 SFO   VX             2179                 588                27.0
##  61 BWI   WN              200                  54                27
##  62 TPA   DL             2111                 571                27.0
##  63 IND   9E              379                 103                27.2
##  64 LAX   AA             3546                 984                27.7
```

```
##  65 RSW   UA              1061              296              27.9
##  66 MCO   DL              3642             1017              27.9
##  67 BOS   UA              3297              937              28.4
##  68 MTJ   UA                14                4              28.6
##  69 ROC   9E               273               78              28.6
##  70 DFW   AA              6966             1995              28.6
##  71 MSP   DL              2829              813              28.7
##  72 LAS   UA              1990              577              29.0
##  73 CHS   B6               612              178              29.1
##  74 SEA   AA               360              105              29.2
##  75 BOS   9E               853              249              29.2
##  76 BNA   WN              1273              373              29.3
##  77 PWM   DL               233               69              29.6
##  78 CVG   9E              1466              436              29.7
##  79 CHS   9E               331               99              29.9
##  80 SJU   AA              1085              325              30.0
##  81 SBN   EV                10                3              30
##  82 BWI   MQ               333              100              30.0
##  83 PHX   UA              1111              334              30.1
##  84 MSY   DL              1114              335              30.1
##  85 BOS   B6              4325             1301              30.1
##  86 PBI   DL              1450              439              30.3
##  87 PIT   9E               820              249              30.4
##  88 CLT   US              8498             2604              30.6
##  89 TVC   EV                62               19              30.6
##  90 LAX   VX              2554              786              30.8
##  91 IAD   9E               612              189              30.9
##  92 CLE   UA              1863              578              31.0
##  93 MIA   DL              2902              902              31.1
##  94 BDL   EV               405              126              31.1
##  95 TPA   AA               305               95              31.1
##  96 BUF   9E               789              246              31.2
##  97 HNL   UA               359              112              31.2
##  98 LGB   B6               661              207              31.3
##  99 STT   UA               188               59              31.4
## 100 DTW   MQ              1852              587              31.7
## 101 DFW   UA              1068              339              31.7
## 102 FLL   DL              2876              913              31.7
## 103 ACK   B6               264               84              31.8
## 104 PIT   DL               247               79              32.0
## 105 PIT   EV              1232              395              32.1
## 106 SYR   9E               159               51              32.1
## 107 SJC   B6               327              105              32.1
## 108 TPA   UA              1952              631              32.3
## 109 ORD   UA              6744             2195              32.5
## 110 PHX   US              2240              731              32.6
## 111 BTV   B6              1347              440              32.7
## 112 PWM   B6              1285              420              32.7
## 113 MYR   EV                58               19              32.8
## 114 BUF   B6              2773              914              33.0
## 115 PDX   DL               457              151              33.0
## 116 RSW   DL               416              138              33.2
## 117 PDX   UA               563              187              33.2
## 118 AUS   UA               664              221              33.3
```

```
## 119 CAE   9E                3                 1                33.3
## 120 ATL   UA              102                34                33.3
## 121 CLT   YV              258                86                33.3
## 122 BWI   9E              815               272                33.4
## 123 ATL   DL            10452              3492                33.4
## 124 ORF   9E              374               125                33.4
## 125 SAT   DL              302               101                33.4
## 126 SJU   UA              685               230                33.6
## 127 SFO   B6             1020               345                33.8
## 128 IAH   UA             6814              2305                33.8
## 129 CMH   EV              774               262                33.9
## 130 SYR   B6             1249               424                33.9
## 131 DTW   9E              954               324                34.0
## 132 IAD   B6              667               228                34.2
## 133 ROC   B6             1388               475                34.2
## 134 SAN   UA             1125               386                34.3
## 135 GSP   9E               96                33                34.4
## 136 MCI   9E              334               115                34.4
## 137 LAX   UA             5770              1989                34.5
## 138 DCA   US             4468              1543                34.5
## 139 SAN   B6              660               228                34.5
## 140 SFO   UA             6728              2326                34.6
## 141 LAX   B6             1669               578                34.6
## 142 SAT   UA              323               112                34.7
## 143 CHO   EV               46                16                34.8
## 144 BUF   EV             1005               350                34.8
## 145 JAX   9E              373               130                34.9
## 146 DEN   UA             3737              1303                34.9
## 147 OAK   B6              309               108                35.0
## 148 TYS   9E              265                93                35.1
## 149 PBI   UA             1825               641                35.1
## 150 JAX   B6             1013               356                35.1
## 151 LAS   AA              634               223                35.2
## 152 BGR   EV              358               126                35.2
## 153 RDU   B6              789               279                35.4
## 154 MKE   WN             1275               451                35.4
## 155 BNA   9E              452               160                35.4
## 156 ILM   EV              107                38                35.5
## 157 SRQ   B6              833               297                35.7
## 158 SEA   UA             1101               393                35.7
## 159 HDN   UA               14                 5                35.7
## 160 SYR   EV              299               107                35.8
## 161 AVL   EV              251                90                35.9
## 162 CLE   EV              555               200                36.0
## 163 MSP   9E             1203               435                36.2
## 164 ORF   MQ              345               125                36.2
## 165 TVC   MQ               33                12                36.4
## 166 FLL   UA             2376               865                36.4
## 167 LAS   B6             1303               475                36.5
## 168 MSY   9E              410               150                36.6
## 169 BQN   B6              593               217                36.6
## 170 PHL   US              623               228                36.6
## 171 CLE   MQ             1634               600                36.7
## 172 STL   WN             1448               533                36.8
```

```
## 173 MSP   EV             1661              612             36.8
## 174 MIA   UA             1548              573             37.0
## 175 DEN   DL             1036              384             37.1
## 176 BZN   UA               35               13             37.1
## 177 RDU   EV             1579              587             37.2
## 178 MCO   AA              725              270             37.2
## 179 BTV   EV             1161              433             37.3
## 180 HOU   WN             1372              512             37.3
## 181 SEA   B6              513              192             37.4
## 182 RDU   9E              865              324             37.5
## 183 ANC   UA                8                3             37.5
## 184 ATL   9E               56               21             37.5
## 185 IND   EV             1245              467             37.5
## 186 ORF   EV              715              269             37.6
## 187 RSW   B6             1958              737             37.6
## 188 SRQ   9E               77               29             37.7
## 189 RDU   MQ             4536             1710             37.7
## 190 BUR   B6              370              140             37.8
## 191 DCA   MQ             2063              783             38.0
## 192 SJU   B6             2709             1031             38.1
## 193 DTW   EV             2389              918             38.4
## 194 IAD   EV             3824             1473             38.5
## 195 HOU   B6              711              274             38.5
## 196 ABQ   B6              254               98             38.6
## 197 MSY   B6             1064              412             38.7
## 198 CMH   MQ             2541              989             38.9
## 199 MSY   EV              566              221             39.0
## 200 CLT   B6              721              282             39.1
## 201 ORD   9E              984              386             39.2
## 202 ROC   EV              697              274             39.3
## 203 MHT   EV              922              364             39.5
## 204 IND   MQ              352              139             39.5
## 205 ALB   EV              418              166             39.7
## 206 MDW   WN             4025             1604             39.9
## 207 RIC   9E              321              128             39.9
## 208 SFO   AA             1398              559             40.0
## 209 MKE   FL               55               22             40
## 210 PHL   9E              860              344             40
## 211 XNA   MQ              708              284             40.1
## 212 BHM   EV              269              108             40.1
## 213 PSE   B6              358              144             40.2
## 214 MCO   B6             6409             2586             40.3
## 215 DSM   9E               84               34             40.5
## 216 GSO   EV             1491              605             40.6
## 217 DFW   9E              346              141             40.8
## 218 CHS   EV             1815              744             41.0
## 219 ORD   B6              892              366             41.0
## 220 FLL   B6             6466             2670             41.3
## 221 SAV   EV              749              310             41.4
## 222 TPA   MQ              250              104             41.6
## 223 PHL   EV               48               20             41.7
## 224 BNA   EV             2054              861             41.9
## 225 ORD   MQ             2097              881             42.0
## 226 OMA   EV              814              342             42.0
```

```
## 227 BNA   MQ              2304              978              42.4
## 228 AUS   B6               742              315              42.5
## 229 MSY   WN               296              126              42.6
## 230 XNA   EV               284              121              42.6
## 231 BDL   UA                 7                3              42.9
## 232 PIT   MQ               356              153              43.0
## 233 MEM   EV              1259              543              43.1
## 234 BQN   UA               295              128              43.4
## 235 MSP   MQ              1230              534              43.4
## 236 DEN   B6               336              146              43.5
## 237 PVD   EV               358              157              43.9
## 238 CLT   9E               271              119              43.9
## 239 DEN   WN              1379              606              43.9
## 240 SDF   EV               851              374              43.9
## 241 TPA   B6              2768             1217              44.0
## 242 CRW   MQ               134               59              44.0
## 243 MKE   9E               297              131              44.1
## 244 PBI   B6              3126             1380              44.1
## 245 GSP   EV               694              308              44.4
## 246 STL   EV              1693              753              44.5
## 247 CVG   EV              1905              848              44.5
## 248 CLT   MQ              1550              690              44.5
## 249 SAN   AA               357              159              44.5
## 250 AUS   AA               358              160              44.7
## 251 PIT   B6                89               40              44.9
## 252 SLC   B6               364              164              45.1
## 253 MSN   EV               555              252              45.4
## 254 DSM   EV               439              200              45.6
## 255 DAY   EV              1057              483              45.7
## 256 BWI   EV               339              156              46.0
## 257 GRR   EV               690              319              46.2
## 258 CAK   FL               842              392              46.6
## 259 PDX   B6               322              150              46.6
## 260 MCI   EV              1471              686              46.6
## 261 PHX   B6               363              170              46.8
## 262 CLT   EV              2374             1117              47.1
## 263 SMF   B6               282              133              47.2
## 264 RIC   EV              2025              959              47.4
## 265 EGE   AA               101               48              47.5
## 266 PHX   WN               425              203              47.8
## 267 CLE   9E               321              155              48.3
## 268 PWM   EV               770              372              48.3
## 269 IAH   AA               271              131              48.3
## 270 JAX   EV              1236              599              48.5
## 271 STL   MQ               134               65              48.5
## 272 ATL   MQ              2235             1089              48.7
## 273 ATL   EV              1656              811              49.0
## 274 AUS   9E                 2                1              50
## 275 DCA   UA                 2                1              50
## 276 DTW   OO                 2                1              50
## 277 ORD   EV                 2                1              50
## 278 PIT   UA                 2                1              50
## 279 STL   UA                 2                1              50
## 280 CVG   DL                 4                2              50
```

```
## 281 ATL   WN              58              29              50
## 282 MKE   EV            1082             545            50.4
## 283 DCA   EV            1565             797            50.9
## 284 DEN   F9             681             351            51.5
## 285 IAD   YV             278             144            51.8
## 286 EYW   DL              17               9            52.9
## 287 ATL   FL            2278            1210            53.1
## 288 CVG   MQ             350             188            53.7
## 289 TYS   EV             313             182            58.1
## 290 OKC   EV             315             189            60
## 291 TUL   EV             294             179            60.9
## 292 CMH   9E              11               7            63.6
## 293 JAC   UA              19              14            73.7
## 294 CAE   EV             103              80            77.7
## 295 CLT   UA               2               2           100
## 296 JAC   DL               2               2           100
## 297 PBI   EV               6               6           100
```

4 Sección 6. Análisis exploratorio de nuestros datos: Lecciones 53 a 66.

4.1 Tarea 12: Introducción a la exloarción de datos. Lecciones 53 a 60

Vamos a repasar el análisis de la variación con lo aprendido en la sección. Intenta hacer los análisis de forma tan detallada como te sea posible, generando preguntas con sentido y intentando obtener una respuesta coherente.

Preguntas de esta tarea

4.1.1 Cuestión 12.1.

Explora la distribución de las variables `x`, `y`, `z` del dataset de `diamonds`. ¿Qué podemos inferir?

Busca un diamante (por internet por ejemplo) y decide qué dimensiones pueden ser aceptables para las medidas de longitud, altura y anchura de un diamante.

4.1.2 Cuestión 12.2.

Explora la distribución del precio (`price`) del dataset de `diamonds`. ¿Hay algo que te llame la atención o resulte un poco extraño?

Recuerda hacer uso del parámetro `binwidth` para probar un rango dispar de valores hasta ver algo que te llame la atención.

4.1.3 Cuestión 12.3.

¿Cuantos diamantes hay de 0.99 quilates? ¿Y de exactamente 1 quilate?

¿A qué puede ser debida esta diferencia?

4.1.4 Cuestión 12.4.

Compara y contrasta el uso de las funciones `coord_cartesian()` frente `xlim()` y `ylim()` para hacer *zoom* en un histograma.

¿Qué ocurre si dejamos el parámetro `binwidth` sin configurar?

¿Qué ocurre si hacemos *zoom* y solamente se ve media barra?

4.1.5 Cuestión 12.5.

- ¿Qué ocurre cuando hay NAs en un histograma?
- ¿Qué ocurre cuando hay NAs en un diagrama de barras?
- ¿Qué diferencias observas?

4.1.6 Cuestión 12.6.

¿Qué hace la opción `na.rm = TRUE` en las funciones `mean()` y `sum()`?

4.2 Tarea 13: Visualización de la covarianza entre variables. Lecciones 61 a 65.

Repasa todo lo aprendido acerca de boxplots, densidades, mapas de calor... porque es hora de extraer más información acerca de los datos de nuestros datasets.

Preguntas de esta tarea

4.2.1 Cuestión 13.1.

Es hora de aplicar todo lo que hemos aprendido para visualizar mejor los tiempos de salida para vuelos cancelados vs los no cancelados. Recuerda bien qué tipo de dato tenemos en cada caso. ¿Qué deduces acerca de los retrasos según la hora del día a la que está programada el vuelo de salida?

4.2.2 Cuestión 13.2.

1. ¿Qué variable del dataset de diamantes crees que es la más importante para poder predecir el precio de un diamante?
2. ¿Qué variable del dataset de diamantes crees que es la que más correlacionada está con cut?
3. ¿Por qué combinar estas dos variables nos lleva a que los diamantes con peor calidad son los mas caros?

4.2.3 Cuestión 13.3.

Instala el paquete de ggstance y úsalo para crear un boxplot horizontal. Compara el resultado con usar el coord_flip() que hemos visto en clase.

4.2.4 Cuestión 13.4.

Los boxplots nacen en una época donde los datasets eran mucho más pequeños y la palabra big data no era más que un concepto futurista. De ahí que los datos considerados con outliers tuvieran sentido que fueran representados con puntos dado que su existencia era más bien escasa o nula. Para solucionar este problema, existe el letter value plot del paquete lvplot. Instala dicho paquete y usa la geometría geom_lv() para mostrar la distribución de precio vs cut de los diamantes. ¿Qué observas y qué puedes interpretar a raíz de dicho gráfico?

4.2.5 Cuestión 13.5.

Compara el uso de la geometría geom_violin() con un facet de geom_histogram() y contra un geom_freqpoly() coloreado. Investiga cuales son los pros y los contras de cada uno de los tipos de representación.

4.2.6 Cuestión 13.6.

Si tenemos datasets pequeños, a veces es útil usar la opción que ya conocemos de geom_jitter() para ver la relación entre una variable contínua y una variable categórica. El paquete de R ggbeeswarm tiene un par de métodos similares a geom_jitter() que te pueden ayudar a tal efecto. Listalos y haz un gráfico con cada uno de ellos para ver qué descripción de los datos podemos extraer de cada uno. ¿A qué gráfico de los que ya has visto durante esta práctica se parece?

4.2.7 Cuestión 13.7.

Los mapas de calor que hemos visto tienen un claro problema de elección de los colores.

```
* ¿Cómo podríamos reescalar el campo count dataset de diamantes cuando cruzamos color y cut para observa
* ¿Por qué resulta mejor usar la estética aes(x = color, y = cut) en lugar de aes(x=cut, y = color)?
```

4.2.8 Cuestión 13.8.

Utiliza la geom_tile() junto con dplyr para explorar si el promedio del retraso de los vuelos varía con respecto al destino y mes del año.

¿Qué hace que este gráfico sea dificil de leer o de interpretar? ¿Cómo puedes mejorar la visualización?

4.2.9 Cuestión 13.9.

En lugar de hacer un resumen de la distribución condicional de dos variables numéricas con un boxplot, se puede usar un polígono de frecuencias.

- ¿Qué hay que tener en cuenta cuando usas cut_width() o cuando usas cut_number()?
- ¿Cómo influye este hecho en la visualización 2D de carat y price
- Da la mejor visualización posible de carat dividido por price.

4.2.10 Cuestión 13.10.

Compara la distribución del precio de los diamantes grandes vs diamantes pequeños. Elige el concepto de grande y pequeño que consideres. Comenta el resultado.

4.2.11 Cuestión 13.11.

Combina diferentes técnicas de ggplot para visulaizar la distribución combinada de cut, carat y precio.

4.2.12 Cuestión 13.12.

Los plots en 2D pueden revelar outliers que no se ven en plots de una sola dimensión. Por ejemplo, algunos puntos del plot dado por

```
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = x, y = y)) +
  coord_cartesian(xlim = c(4,12), ylim = c(4,12))
```

hacen destacar muchísimo los outliers combinando x con y, a pesar de que por separado parecen valores normales.

Intenta averiguar porqué un scatterplot resulta más efectivo en este caso que un gráfico con agrupaciones.

5 Enunciado taller entregable 2

Consideremos las siguiente preguntas tres del taller de clase 11. Se muestra una solución parcial a cada pregunta.

5.0.1 Cuestión 11-10

Los retrasos suelen estar correlacionados con el tiempo. Aunque el problema que ha causado el primer retraso de un avión se resuelva, el resto de vuelos se retrasan para que salgan primero los aviones que debían haber partido antes. Intenta usar la función `lag()` explora cómo el retraso de un avión se relaciona con el retraso del avión inmediatamente anterior o posterior.

5.0.2 Solución

Seleccionaremos las salidas de JFK y los retrasos en la salida. Necesitamos ordenar los vuelos por salida

```
flights %>% filter(origin == "JFK") %>%
  transmute(origin, time_hour, dep_delay) %>%
  arrange(time_hour) -> flights_time_order


  lapply(1:10, function(x)
  cor(
  flights_time_order$dep_delay,
  lag(flights_time_order$dep_delay, x),
  use = "complete.obs"
  )) -> correlation_delay_lag

  correlation_delay_lag <- unlist(correlation_delay_lag)
  names(correlation_delay_lag) <- paste("lag", 1:10, sep = "_")
  correlation_delay_lag
```

```
##      lag_1      lag_2      lag_3      lag_4      lag_5      lag_6
## 0.55227212 0.38265103 0.28005542 0.21765785 0.17309678 0.14035150
##      lag_7      lag_8      lag_9     lag_10
## 0.11570897 0.09918925 0.09461354 0.08716629
```

Vemos como las correlaciones entre los retrasos decrecen

5.0.3 Cuestión 11-11

Vamos a por los destinos esta vez. Localiza vuelos que llegaron 'demasiado rápido' a sus destinos. Seguramente, el becario se equivocó al introducir el tiempo de vuelo y se trate de un error en los datos. Calcula para ello el cociente entre el tiempo en el aire de cada vuelo relativo al tiempo de vuelo del avión que tardó menos en llegar a dicho destino. ¿Qué vuelos fueron los que más se retrasaron en el aire?

5.0.4 Solución

```
flights %>% filter(origin=="JFK",!is.na(air_time))  -> JFK_fli

JFK_fli %>% right_join( JFK_fli %>% group_by(dest) %>%
                          summarise(min_air_time=min(air_time),key="dest")) %>%
  transmute(dest,tailnum,carrier,sched_dep_time,sched_arr_time,air_time,
            min_air_time,rel_air_time_by_dest=air_time/min_air_time) %>%
  arrange(desc(rel_air_time_by_dest))-> JFK_fli
```

```
## Joining, by = "dest"
```

```
JFK_fli
```

```
## # A tibble: 109,079 x 8
##    dest  tailnum carrier sched_dep_time sched_arr_time air_time
##    <chr> <chr>   <chr>            <int>          <int>    <dbl>
##  1 ACK   N328JB  B6                 800            909      141
##  2 BOS   N3FKAA  AA                1605           1740       96
##  3 BOS   N346NB  DL                1200           1317       91
##  4 BOS   N913XJ  9E                 840           1003       86
##  5 BOS   N3DRAA  AA                1245           1355       86
##  6 BOS   N3FEAA  AA                1245           1350       80
```

```
##  7 BOS   N279JB  B6                1645           1813       77
##  8 BOS   N3FJAA  AA                1600           1720       76
##  9 PHL   N8932C  9E                 940           1051       61
## 10 DCA   N813MQ  MQ                 755            910       97
## # ... with 109,069 more rows, and 2 more variables: min_air_time <dbl>,
## #   rel_air_time_by_dest <dbl>
```

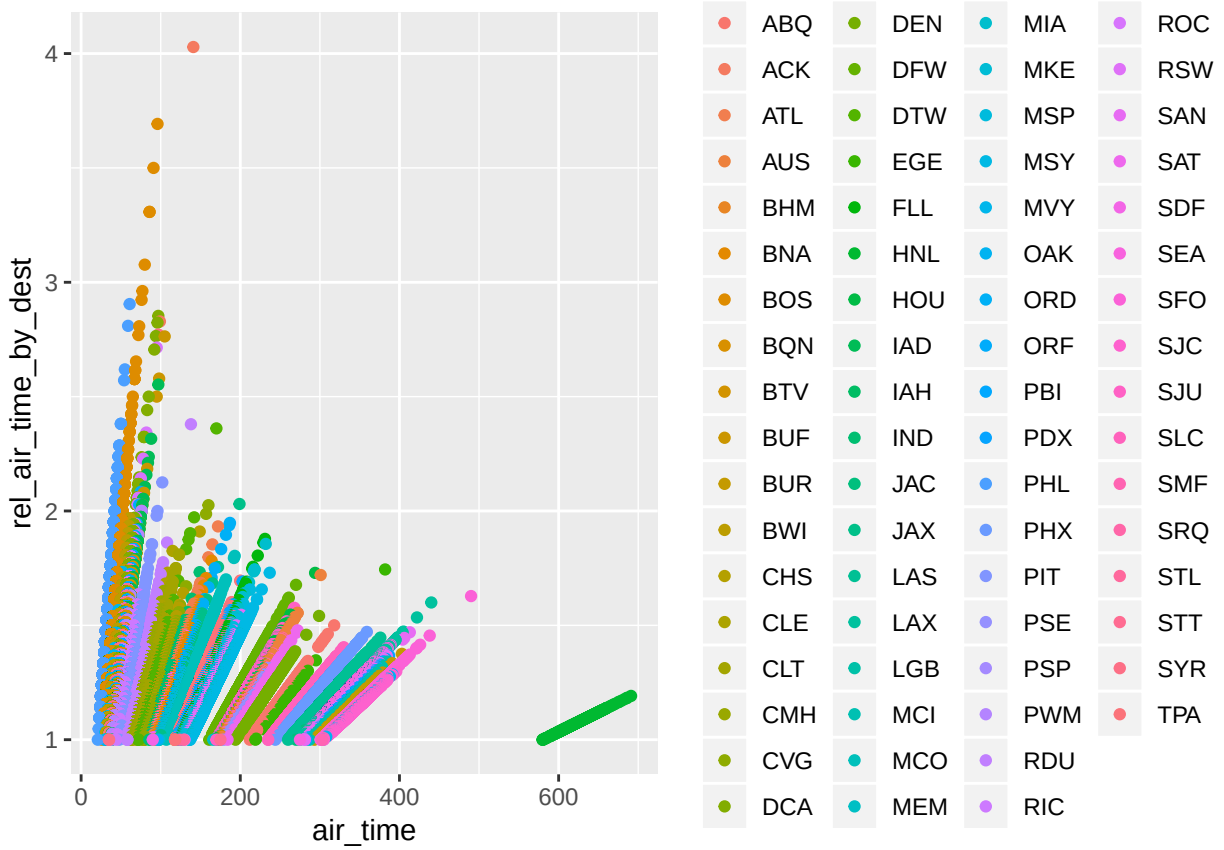
Los vuelos a los aeropuertos con más retraso relativo son de los aeropuertos

```
airports[airports$faa %in% c("ACK","BOS","DCA"),]
```

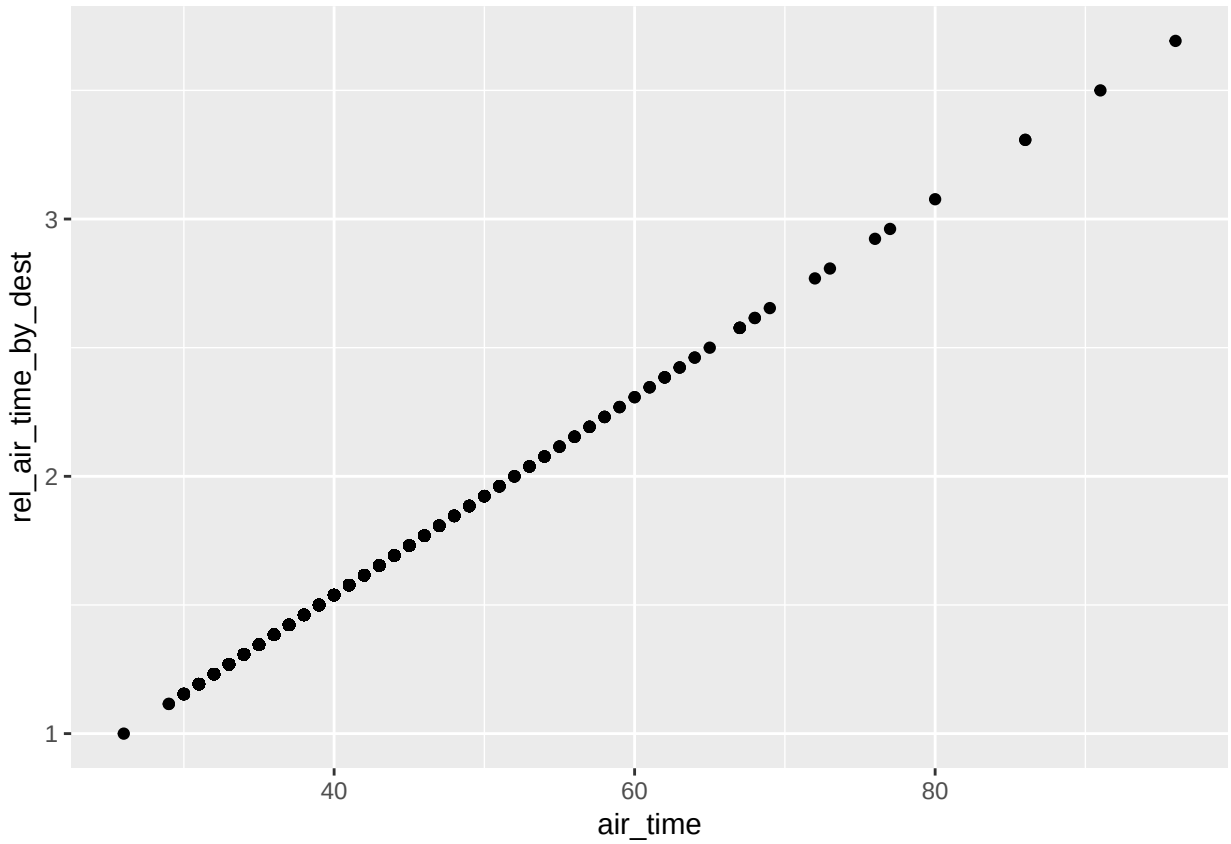
```
## # A tibble: 3 x 8
##   faa   name                          lat   lon   alt    tz dst   tzone
##   <chr> <chr>                       <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1 ACK   Nantucket Mem                41.3 -70.1    48    -5 A     America/New~
## 2 BOS   General Edward Lawrence~     42.4 -71.0    19    -5 A     America/New~
## 3 DCA   Ronald Reagan Washingto~     38.9 -77.0    15    -5 A     America/New~
```

Un gráfico de los tiempos relativos contra los absolutos (no interpretéis mucho las rectas pues lo son y=x/min), el gráfico es mejorable.

```
JFK_fli %>% ggplot(aes(x=air_time,y=rel_air_time_by_dest))+ geom_point(aes(col=dest))
```




```
JFK_fli %>% filter(dest=="BOS")%>%
 ggplot(aes(x=air_time,y=rel_air_time_by_dest))+ geom_point()
```



5.0.5 Cuestión 11-12

Encuentra todos los destinos a los que vuelan dos o más compañías y para cada uno de ellos, crea un ranking de las mejores compañías para volar a cada destino (utiliza el criterio que consideres más conveniente como probabilidad de retraso, velocidad o tiempo de vuelo, número de vuelos al año..)

Finalmente, para cada avión (basándonos en el número de cola) cuenta el número de vuelos que hace antes de sufrir su primer retraso de más de una hora. Valora entonces la fiabilidad del avión o de la compañía aérea asociada al mismo.

5.0.6 Solución

```
flights %>% group_by(dest, carrier) %>% count() %>% filter(n >= 2) -> aux

filter_dest_carrier = paste(aux$dest, aux$carrier)
filter_dest_carrier
```

```
##   [1] "ABQ B6" "ACK B6" "ALB EV" "ANC UA" "ATL 9E" "ATL DL" "ATL EV"
##   [8] "ATL FL" "ATL MQ" "ATL UA" "ATL WN" "AUS 9E" "AUS AA" "AUS B6"
##  [15] "AUS DL" "AUS UA" "AUS WN" "AVL 9E" "AVL EV" "BDL EV" "BDL UA"
##  [22] "BGR EV" "BHM EV" "BNA 9E" "BNA EV" "BNA MQ" "BNA WN" "BOS 9E"
##  [29] "BOS AA" "BOS B6" "BOS DL" "BOS EV" "BOS UA" "BOS US" "BQN B6"
##  [36] "BQN UA" "BTV 9E" "BTV B6" "BTV EV" "BUF 9E" "BUF B6" "BUF DL"
##  [43] "BUF EV" "BUR B6" "BWI 9E" "BWI EV" "BWI MQ" "BWI WN" "BZN UA"
##  [50] "CAE 9E" "CAE EV" "CAK FL" "CHO EV" "CHS 9E" "CHS B6" "CHS EV"
##  [57] "CLE 9E" "CLE EV" "CLE MQ" "CLE OO" "CLE UA" "CLT 9E" "CLT B6"
##  [64] "CLT EV" "CLT MQ" "CLT UA" "CLT US" "CLT YV" "CMH 9E" "CMH EV"
##  [71] "CMH MQ" "CRW MQ" "CVG 9E" "CVG DL" "CVG EV" "CVG MQ" "DAY 9E"
```

```
##  [78] "DAY EV" "DCA 9E" "DCA DL" "DCA EV" "DCA MQ" "DCA UA" "DCA US"
##  [85] "DEN B6" "DEN DL" "DEN F9" "DEN UA" "DEN WN" "DFW 9E" "DFW AA"
##  [92] "DFW EV" "DFW UA" "DSM 9E" "DSM EV" "DTW 9E" "DTW DL" "DTW EV"
##  [99] "DTW MQ" "DTW OO" "EGE AA" "EGE UA" "EYW DL" "FLL AA" "FLL B6"
## [106] "FLL DL" "FLL UA" "GRR 9E" "GRR EV" "GSO EV" "GSP 9E" "GSP EV"
## [113] "HDN UA" "HNL HA" "HNL UA" "HOU B6" "HOU WN" "IAD 9E" "IAD B6"
## [120] "IAD EV" "IAD YV" "IAH AA" "IAH UA" "ILM EV" "IND 9E" "IND DL"
## [127] "IND EV" "IND MQ" "IND UA" "JAC DL" "JAC UA" "JAX 9E" "JAX B6"
## [134] "JAX EV" "LAS AA" "LAS B6" "LAS DL" "LAS UA" "LAS VX" "LAX AA"
## [141] "LAX B6" "LAX DL" "LAX UA" "LAX VX" "LGB B6" "MCI 9E" "MCI DL"
## [148] "MCI EV" "MCO AA" "MCO B6" "MCO DL" "MCO UA" "MDW WN" "MEM 9E"
## [155] "MEM DL" "MEM EV" "MHT 9E" "MHT EV" "MIA AA" "MIA DL" "MIA UA"
## [162] "MKE 9E" "MKE EV" "MKE FL" "MKE WN" "MSN EV" "MSP 9E" "MSP DL"
## [169] "MSP EV" "MSP MQ" "MSP OO" "MSP UA" "MSY 9E" "MSY B6" "MSY DL"
## [176] "MSY EV" "MSY UA" "MSY WN" "MTJ UA" "MVY 9E" "MVY B6" "MYR EV"
## [183] "OAK B6" "OKC EV" "OMA EV" "OMA UA" "ORD 9E" "ORD AA" "ORD B6"
## [190] "ORD EV" "ORD MQ" "ORD UA" "ORF 9E" "ORF EV" "ORF MQ" "PBI AA"
## [197] "PBI B6" "PBI DL" "PBI EV" "PBI UA" "PDX B6" "PDX DL" "PDX UA"
## [204] "PHL 9E" "PHL DL" "PHL EV" "PHL US" "PHL YV" "PHX B6" "PHX DL"
## [211] "PHX UA" "PHX US" "PHX WN" "PIT 9E" "PIT B6" "PIT DL" "PIT EV"
## [218] "PIT MQ" "PIT UA" "PSE B6" "PSP VX" "PVD EV" "PWM B6" "PWM DL"
## [225] "PWM EV" "RDU 9E" "RDU B6" "RDU EV" "RDU MQ" "RIC 9E" "RIC EV"
## [232] "ROC 9E" "ROC B6" "ROC EV" "RSW 9E" "RSW B6" "RSW DL" "RSW UA"
## [239] "SAN AA" "SAN B6" "SAN DL" "SAN UA" "SAT 9E" "SAT DL" "SAT UA"
## [246] "SAV EV" "SBN EV" "SDF 9E" "SDF EV" "SDF UA" "SEA AA" "SEA AS"
## [253] "SEA B6" "SEA DL" "SEA UA" "SFO AA" "SFO B6" "SFO DL" "SFO UA"
## [260] "SFO VX" "SJC B6" "SJU AA" "SJU B6" "SJU DL" "SJU UA" "SLC B6"
## [267] "SLC DL" "SMF B6" "SNA UA" "SRQ 9E" "SRQ B6" "SRQ DL" "SRQ EV"
## [274] "STL AA" "STL EV" "STL MQ" "STL UA" "STL WN" "STT AA" "STT DL"
## [281] "STT UA" "SYR 9E" "SYR B6" "SYR EV" "TPA 9E" "TPA AA" "TPA B6"
## [288] "TPA DL" "TPA MQ" "TPA UA" "TUL EV" "TVC EV" "TVC MQ" "TYS 9E"
## [295] "TYS EV" "XNA EV" "XNA MQ"
```

```
flights %>% filter(paste(dest, carrier)%in%
                     filter_dest_carrier&!is.na(arr_delay)) %>%
  transmute(dest, carrier, arr_delay, flights_delayeds = arr_delay >= 5)%>%
  group_by(dest, carrier) %>%
  summarise(
    total_flights = n(),
    total_flights_delayeds = sum(flights_delayeds),
    percent_flights_delayeds = 100 * total_flights_delayeds / total_flights
  ) %>%
  arrange(percent_flights_delayeds, total_flights) %>%
  print(pp, n = Inf, width = Inf)
```

```
## # A tibble: 297 x 5
## # Groups:   dest [103]
##     dest  carrier total_flights total_flights_delayeds
##     <chr> <chr>           <int>                  <int>
##   1 BTV   9E                  2                      0
##   2 DCA   DL                  2                      0
##   3 IND   DL                  2                      0
##   4 MSP   UA                  2                      0
##   5 OMA   UA                  2                      0
##   6 PHL   DL                  2                      0
```

```
##   7 BUF   DL                3                   0
##   8 IND   UA                3                   0
##   9 MEM   9E                3                   0
##  10 SDF   UA                3                   0
##  11 TPA   9E                3                   0
##  12 MHT   9E               10                   0
##  13 STT   DL               29                   1
##  14 DFW   EV                8                   1
##  15 PBI   AA               80                  11
##  16 AUS   WN              293                  42
##  17 PSP   VX               18                   3
##  18 SRQ   EV               28                   5
##  19 BOS   EV              156                  29
##  20 AVL   9E               10                   2
##  21 MCI   DL               80                  16
##  22 SRQ   DL              263                  53
##  23 SEA   DL             1202                 248
##  24 LAS   VX              364                  76
##  25 RSW   9E               67                  14
##  26 SJU   DL             1294                 274
##  27 MVY   9E               65                  14
##  28 BOS   US             4002                 879
##  29 SEA   AS              709                 159
##  30 HNL   HA              342                  78
##  31 DAY   9E              342                  79
##  32 BOS   DL              961                 222
##  33 FLL   AA              179                  42
##  34 EGE   UA              106                  25
##  35 DTW   DL             3833                 909
##  36 CLE   OO               21                   5
##  37 SFO   DL             1848                 443
##  38 SDF   9E              250                  60
##  39 MVY   B6              145                  35
##  40 MSY   UA              265                  64
##  41 STT   AA              301                  74
##  42 SNA   UA              812                 200
##  43 MSP   OO                4                   1
##  44 PHL   YV                8                   2
##  45 PHX   DL              467                 117
##  46 MIA   AA             7143                1798
##  47 STL   AA              864                 219
##  48 BOS   AA             1428                 362
##  49 LAS   DL             1661                 434
##  50 AUS   DL              352                  92
##  51 SLC   DL             2087                 548
##  52 GRR   9E               38                  10
##  53 SAT   9E               34                   9
##  54 MCO   UA             3191                 852
##  55 ORD   AA             5846                1562
##  56 LAX   DL             2487                 667
##  57 MEM   DL              424                 114
##  58 DCA   9E             1011                 272
##  59 SAN   DL              567                 153
##  60 SFO   VX             2179                 588
```

```
##  61 BWI   WN              200                54
##  62 TPA   DL             2111               571
##  63 IND   9E              379               103
##  64 LAX   AA             3546               984
##  65 RSW   UA             1061               296
##  66 MCO   DL             3642              1017
##  67 BOS   UA             3297               937
##  68 MTJ   UA               14                 4
##  69 ROC   9E              273                78
##  70 DFW   AA             6966              1995
##  71 MSP   DL             2829               813
##  72 LAS   UA             1990               577
##  73 CHS   B6               612               178
##  74 SEA   AA               360               105
##  75 BOS   9E               853               249
##  76 BNA   WN              1273               373
##  77 PWM   DL               233                69
##  78 CVG   9E              1466               436
##  79 CHS   9E               331                99
##  80 SJU   AA              1085               325
##  81 SBN   EV                10                 3
##  82 BWI   MQ               333               100
##  83 PHX   UA              1111               334
##  84 MSY   DL              1114               335
##  85 BOS   B6              4325              1301
##  86 PBI   DL              1450               439
##  87 PIT   9E               820               249
##  88 CLT   US              8498              2604
##  89 TVC   EV                62                19
##  90 LAX   VX              2554               786
##  91 IAD   9E               612               189
##  92 CLE   UA              1863               578
##  93 MIA   DL              2902               902
##  94 BDL   EV               405               126
##  95 TPA   AA               305                95
##  96 BUF   9E               789               246
##  97 HNL   UA               359               112
##  98 LGB   B6               661               207
##  99 STT   UA               188                59
## 100 DTW   MQ              1852               587
## 101 DFW   UA              1068               339
## 102 FLL   DL              2876               913
## 103 ACK   B6               264                84
## 104 PIT   DL               247                79
## 105 PIT   EV              1232               395
## 106 SYR   9E               159                51
## 107 SJC   B6               327               105
## 108 TPA   UA              1952               631
## 109 ORD   UA              6744              2195
## 110 PHX   US              2240               731
## 111 BTV   B6              1347               440
## 112 PWM   B6              1285               420
## 113 MYR   EV                58                19
## 114 BUF   B6              2773               914
```

```
## 115 PDX   DL              457              151
## 116 RSW   DL              416              138
## 117 PDX   UA              563              187
## 118 AUS   UA              664              221
## 119 CAE   9E                3                1
## 120 ATL   UA              102               34
## 121 CLT   YV              258               86
## 122 BWI   9E              815              272
## 123 ATL   DL            10452             3492
## 124 ORF   9E              374              125
## 125 SAT   DL              302              101
## 126 SJU   UA              685              230
## 127 SFO   B6             1020              345
## 128 IAH   UA             6814             2305
## 129 CMH   EV              774              262
## 130 SYR   B6             1249              424
## 131 DTW   9E              954              324
## 132 IAD   B6              667              228
## 133 ROC   B6             1388              475
## 134 SAN   UA             1125              386
## 135 GSP   9E               96               33
## 136 MCI   9E              334              115
## 137 LAX   UA             5770             1989
## 138 DCA   US             4468             1543
## 139 SAN   B6              660              228
## 140 SFO   UA             6728             2326
## 141 LAX   B6             1669              578
## 142 SAT   UA              323              112
## 143 CHO   EV               46               16
## 144 BUF   EV             1005              350
## 145 JAX   9E              373              130
## 146 DEN   UA             3737             1303
## 147 OAK   B6              309              108
## 148 TYS   9E              265               93
## 149 PBI   UA             1825              641
## 150 JAX   B6             1013              356
## 151 LAS   AA              634              223
## 152 BGR   EV              358              126
## 153 RDU   B6              789              279
## 154 MKE   WN             1275              451
## 155 BNA   9E              452              160
## 156 ILM   EV              107               38
## 157 SRQ   B6              833              297
## 158 SEA   UA             1101              393
## 159 HDN   UA               14                5
## 160 SYR   EV              299              107
## 161 AVL   EV              251               90
## 162 CLE   EV              555              200
## 163 MSP   9E             1203              435
## 164 ORF   MQ              345              125
## 165 TVC   MQ               33               12
## 166 FLL   UA             2376              865
## 167 LAS   B6             1303              475
## 168 MSY   9E              410              150
```

```
## 169 BQN   B6              593                217
## 170 PHL   US              623                228
## 171 CLE   MQ             1634                600
## 172 STL   WN             1448                533
## 173 MSP   EV             1661                612
## 174 MIA   UA             1548                573
## 175 DEN   DL             1036                384
## 176 BZN   UA               35                 13
## 177 RDU   EV             1579                587
## 178 MCO   AA              725                270
## 179 BTV   EV             1161                433
## 180 HOU   WN             1372                512
## 181 SEA   B6              513                192
## 182 RDU   9E              865                324
## 183 ANC   UA                8                  3
## 184 ATL   9E               56                 21
## 185 IND   EV             1245                467
## 186 ORF   EV              715                269
## 187 RSW   B6             1958                737
## 188 SRQ   9E               77                 29
## 189 RDU   MQ             4536               1710
## 190 BUR   B6              370                140
## 191 DCA   MQ             2063                783
## 192 SJU   B6             2709               1031
## 193 DTW   EV             2389                918
## 194 IAD   EV             3824               1473
## 195 HOU   B6              711                274
## 196 ABQ   B6              254                 98
## 197 MSY   B6             1064                412
## 198 CMH   MQ             2541                989
## 199 MSY   EV              566                221
## 200 CLT   B6              721                282
## 201 ORD   9E              984                386
## 202 ROC   EV              697                274
## 203 MHT   EV              922                364
## 204 IND   MQ              352                139
## 205 ALB   EV              418                166
## 206 MDW   WN             4025               1604
## 207 RIC   9E              321                128
## 208 SFO   AA             1398                559
## 209 MKE   FL               55                 22
## 210 PHL   9E              860                344
## 211 XNA   MQ              708                284
## 212 BHM   EV              269                108
## 213 PSE   B6              358                144
## 214 MCO   B6             6409               2586
## 215 DSM   9E               84                 34
## 216 GSO   EV             1491                605
## 217 DFW   9E              346                141
## 218 CHS   EV             1815                744
## 219 ORD   B6              892                366
## 220 FLL   B6             6466               2670
## 221 SAV   EV              749                310
## 222 TPA   MQ              250                104
```

```
## 223 PHL   EV              48              20
## 224 BNA   EV            2054             861
## 225 ORD   MQ            2097             881
## 226 OMA   EV             814             342
## 227 BNA   MQ            2304             978
## 228 AUS   B6             742             315
## 229 MSY   WN             296             126
## 230 XNA   EV             284             121
## 231 BDL   UA               7               3
## 232 PIT   MQ             356             153
## 233 MEM   EV            1259             543
## 234 BQN   UA             295             128
## 235 MSP   MQ            1230             534
## 236 DEN   B6             336             146
## 237 PVD   EV             358             157
## 238 CLT   9E             271             119
## 239 DEN   WN            1379             606
## 240 SDF   EV             851             374
## 241 TPA   B6            2768            1217
## 242 CRW   MQ             134              59
## 243 MKE   9E             297             131
## 244 PBI   B6            3126            1380
## 245 GSP   EV             694             308
## 246 STL   EV            1693             753
## 247 CVG   EV            1905             848
## 248 CLT   MQ            1550             690
## 249 SAN   AA             357             159
## 250 AUS   AA             358             160
## 251 PIT   B6              89              40
## 252 SLC   B6             364             164
## 253 MSN   EV             555             252
## 254 DSM   EV             439             200
## 255 DAY   EV            1057             483
## 256 BWI   EV             339             156
## 257 GRR   EV             690             319
## 258 CAK   FL             842             392
## 259 PDX   B6             322             150
## 260 MCI   EV            1471             686
## 261 PHX   B6             363             170
## 262 CLT   EV            2374            1117
## 263 SMF   B6             282             133
## 264 RIC   EV            2025             959
## 265 EGE   AA             101              48
## 266 PHX   WN             425             203
## 267 CLE   9E             321             155
## 268 PWM   EV             770             372
## 269 IAH   AA             271             131
## 270 JAX   EV            1236             599
## 271 STL   MQ             134              65
## 272 ATL   MQ            2235            1089
## 273 ATL   EV            1656             811
## 274 AUS   9E               2               1
## 275 DCA   UA               2               1
## 276 DTW   OO               2               1
```

```
## 277 ORD   EV                  2                     1
## 278 PIT   UA                  2                     1
## 279 STL   UA                  2                     1
## 280 CVG   DL                  4                     2
## 281 ATL   WN                 58                    29
## 282 MKE   EV                1082                   545
## 283 DCA   EV                1565                   797
## 284 DEN   F9                 681                   351
## 285 IAD   YV                 278                   144
## 286 EYW   DL                  17                     9
## 287 ATL   FL                2278                  1210
## 288 CVG   MQ                 350                   188
## 289 TYS   EV                 313                   182
## 290 OKC   EV                 315                   189
## 291 TUL   EV                 294                   179
## 292 CMH   9E                  11                     7
## 293 JAC   UA                  19                    14
## 294 CAE   EV                 103                    80
## 295 CLT   UA                   2                     2
## 296 JAC   DL                   2                     2
## 297 PBI   EV                   6                     6
##     percent_flights_delayeds
##                        <dbl>
##   1                     0
##   2                     0
##   3                     0
##   4                     0
##   5                     0
##   6                     0
##   7                     0
##   8                     0
##   9                     0
##  10                     0
##  11                     0
##  12                     0
##  13                     3.45
##  14                    12.5
##  15                    13.8
##  16                    14.3
##  17                    16.7
##  18                    17.9
##  19                    18.6
##  20                    20
##  21                    20
##  22                    20.2
##  23                    20.6
##  24                    20.9
##  25                    20.9
##  26                    21.2
##  27                    21.5
##  28                    22.0
##  29                    22.4
##  30                    22.8
##  31                    23.1
```

```
##  32                  23.1
##  33                  23.5
##  34                  23.6
##  35                  23.7
##  36                  23.8
##  37                  24.0
##  38                  24
##  39                  24.1
##  40                  24.2
##  41                  24.6
##  42                  24.6
##  43                  25
##  44                  25
##  45                  25.1
##  46                  25.2
##  47                  25.3
##  48                  25.4
##  49                  26.1
##  50                  26.1
##  51                  26.3
##  52                  26.3
##  53                  26.5
##  54                  26.7
##  55                  26.7
##  56                  26.8
##  57                  26.9
##  58                  26.9
##  59                  27.0
##  60                  27.0
##  61                  27
##  62                  27.0
##  63                  27.2
##  64                  27.7
##  65                  27.9
##  66                  27.9
##  67                  28.4
##  68                  28.6
##  69                  28.6
##  70                  28.6
##  71                  28.7
##  72                  29.0
##  73                  29.1
##  74                  29.2
##  75                  29.2
##  76                  29.3
##  77                  29.6
##  78                  29.7
##  79                  29.9
##  80                  30.0
##  81                  30
##  82                  30.0
##  83                  30.1
##  84                  30.1
##  85                  30.1
```

```
##  86                  30.3
##  87                  30.4
##  88                  30.6
##  89                  30.6
##  90                  30.8
##  91                  30.9
##  92                  31.0
##  93                  31.1
##  94                  31.1
##  95                  31.1
##  96                  31.2
##  97                  31.2
##  98                  31.3
##  99                  31.4
## 100                  31.7
## 101                  31.7
## 102                  31.7
## 103                  31.8
## 104                  32.0
## 105                  32.1
## 106                  32.1
## 107                  32.1
## 108                  32.3
## 109                  32.5
## 110                  32.6
## 111                  32.7
## 112                  32.7
## 113                  32.8
## 114                  33.0
## 115                  33.0
## 116                  33.2
## 117                  33.2
## 118                  33.3
## 119                  33.3
## 120                  33.3
## 121                  33.3
## 122                  33.4
## 123                  33.4
## 124                  33.4
## 125                  33.4
## 126                  33.6
## 127                  33.8
## 128                  33.8
## 129                  33.9
## 130                  33.9
## 131                  34.0
## 132                  34.2
## 133                  34.2
## 134                  34.3
## 135                  34.4
## 136                  34.4
## 137                  34.5
## 138                  34.5
## 139                  34.5
```

```
## 140                 34.6
## 141                 34.6
## 142                 34.7
## 143                 34.8
## 144                 34.8
## 145                 34.9
## 146                 34.9
## 147                 35.0
## 148                 35.1
## 149                 35.1
## 150                 35.1
## 151                 35.2
## 152                 35.2
## 153                 35.4
## 154                 35.4
## 155                 35.4
## 156                 35.5
## 157                 35.7
## 158                 35.7
## 159                 35.7
## 160                 35.8
## 161                 35.9
## 162                 36.0
## 163                 36.2
## 164                 36.2
## 165                 36.4
## 166                 36.4
## 167                 36.5
## 168                 36.6
## 169                 36.6
## 170                 36.6
## 171                 36.7
## 172                 36.8
## 173                 36.8
## 174                 37.0
## 175                 37.1
## 176                 37.1
## 177                 37.2
## 178                 37.2
## 179                 37.3
## 180                 37.3
## 181                 37.4
## 182                 37.5
## 183                 37.5
## 184                 37.5
## 185                 37.5
## 186                 37.6
## 187                 37.6
## 188                 37.7
## 189                 37.7
## 190                 37.8
## 191                 38.0
## 192                 38.1
## 193                 38.4
```

```
## 194                  38.5
## 195                  38.5
## 196                  38.6
## 197                  38.7
## 198                  38.9
## 199                  39.0
## 200                  39.1
## 201                  39.2
## 202                  39.3
## 203                  39.5
## 204                  39.5
## 205                  39.7
## 206                  39.9
## 207                  39.9
## 208                  40.0
## 209                  40
## 210                  40
## 211                  40.1
## 212                  40.1
## 213                  40.2
## 214                  40.3
## 215                  40.5
## 216                  40.6
## 217                  40.8
## 218                  41.0
## 219                  41.0
## 220                  41.3
## 221                  41.4
## 222                  41.6
## 223                  41.7
## 224                  41.9
## 225                  42.0
## 226                  42.0
## 227                  42.4
## 228                  42.5
## 229                  42.6
## 230                  42.6
## 231                  42.9
## 232                  43.0
## 233                  43.1
## 234                  43.4
## 235                  43.4
## 236                  43.5
## 237                  43.9
## 238                  43.9
## 239                  43.9
## 240                  43.9
## 241                  44.0
## 242                  44.0
## 243                  44.1
## 244                  44.1
## 245                  44.4
## 246                  44.5
## 247                  44.5
```

```
## 248                  44.5
## 249                  44.5
## 250                  44.7
## 251                  44.9
## 252                  45.1
## 253                  45.4
## 254                  45.6
## 255                  45.7
## 256                  46.0
## 257                  46.2
## 258                  46.6
## 259                  46.6
## 260                  46.6
## 261                  46.8
## 262                  47.1
## 263                  47.2
## 264                  47.4
## 265                  47.5
## 266                  47.8
## 267                  48.3
## 268                  48.3
## 269                  48.3
## 270                  48.5
## 271                  48.5
## 272                  48.7
## 273                  49.0
## 274                  50
## 275                  50
## 276                  50
## 277                  50
## 278                  50
## 279                  50
## 280                  50
## 281                  50
## 282                  50.4
## 283                  50.9
## 284                  51.5
## 285                  51.8
## 286                  52.9
## 287                  53.1
## 288                  53.7
## 289                  58.1
## 290                  60
## 291                  60.9
## 292                  63.6
## 293                  73.7
## 294                  77.7
## 295                 100
## 296                 100
## 297                 100
```

6 Preguntas

Entregad en un fichero Rmd y html (y subirlos a la actividad correspondiente) que responda estas preguntas:

6.1 Pregunta 1

Para las cada una de las tres cuestiones:

1. Comentar qué hace cada línea de código. (5 puntos)
2. Utilizar las salidas del código para responder a la cuestión o justificar que no es posible responder a la cuestión planteada. (2 puntos)

6.2 Pregunta 2

Para las cada una de las tres cuestiones introducid alguna mejora en el el código que mejore total o parcialmente las soluciones de las tres cuestiones propuestas. (3 puntos/ uno por cada cuestión.)

Emisiones de CO2 en el mundo.

El siguiente enlace WorldBankCO2 nos da acceso a un conocido data set de THE WORLD BANK. En concreto la versión de este data set es la de de Tableau Open Data Sets una colección de datos del programa Tableau que es un programa para representar gráficas, paneles de control o *dahsboards* y los llamados KPIs.

En esta actividad se trata en primer lugar que entendáis los datos del fichero y que lo leáis de forma directa desde el archivo .xlsx y transforméis en tibbles o data frames de R.

El fichero excel consta de 9 hojas y podemos explorarlo y leerlo con varios paquetes de R. Uno de estos es `readr`

Por ejemplo el siguiente código nos da los nombres de las sheets del fichero

```
library(readxl)
filename="World_Bank_CO2.xlsx"
sheets_names <- readxl::excel_sheets(filename)
sheets_names
```

```
## [1] "About"                    "CO2 (kt) Pivoted"
## [3] "CO2 (kt) RAW DATA"        "CO2 Data Cleaned"
## [5] "CO2 (kt) for Split"       "CO2 for World to Union"
## [7] "CO2 Per Capita RAW DATA"  "CO2 Per Capita (Pivoted)"
## [9] "Metadata - Countries"
```

Ahora podemos leer cada hoja

```
read_excel_allsheets <- function(filename) {
    sheets_names <- readxl::excel_sheets(filename)
    x <- lapply(sheets_names, function(X) readxl::read_excel(filename, sheet = X))
    names(x) <- sheets_names
    return(x)
    }
```

El siguiente código lee todas las sheets del excel y pone cada una en una lista de objetos llamada all_data_CO2. Cada objeto se llama con el nombre de la hoja

```
all_data_CO2=read_excel_allsheets(filename)
class(all_data_CO2)
```

```
## [1] "list"
```

```
lapply(all_data_CO2,FUN=function(x) c(class=paste(class(x),collapse=", "),col_names=paste(names(x),colla
```

```
## $About
##
##
##
## "Data from http://data.worldbank.org/indicator/EN.ATM.CO2E.PC and http://data.worldbank.org/indicato
##
## $`CO2 (kt) Pivoted`
##                                                 class
##                             "tbl_df, tbl, data.frame"
##                                             col_names
## "Country Name, Country Code, Region, Year, CO2 (kt)"
##
## $`CO2 (kt) RAW DATA`
##
##
##
## "Country Name, Country Code, Indicator Name, Indicator Code, 1960, 1961, 1962, 1963, 1964, 1965, 196
##
## $`CO2 Data Cleaned`
##                                                                                      class
##                                                                 "tbl_df, tbl, data.frame"
##                                                                                  col_names
## "Country Code, Country Name, Region, Year, CO2 (kt), CO2 Per Capita (metric tons)"
##
## $`CO2 (kt) for Split`
##                                                 class
##                             "tbl_df, tbl, data.frame"
##                                             col_names
## "Country Code, Country Name, Region, Year, CO2 (kt)"
##
## $`CO2 for World to Union`
##                       class                 col_names
## "tbl_df, tbl, data.frame"                         ""
##
## $`CO2 Per Capita RAW DATA`
##
##
##
## "Country Name, Country Code, Indicator Name, Indicator Code, 1960, 1961, 1962, 1963, 1964, 1965, 196
##
## $`CO2 Per Capita (Pivoted)`
##                                                           class
##                                          "tbl_df, tbl, data.frame"
##                                                         col_names
## "Country Name, Country Code, Year, CO2 Per Capita (metric tons)"
##
## $`Metadata - Countries`
##                                                         class
##                                       "tbl_df, tbl, data.frame"
##                                                       col_names
## "Country Code, Region, IncomeGroup, SpecialNotes, TableName"
```

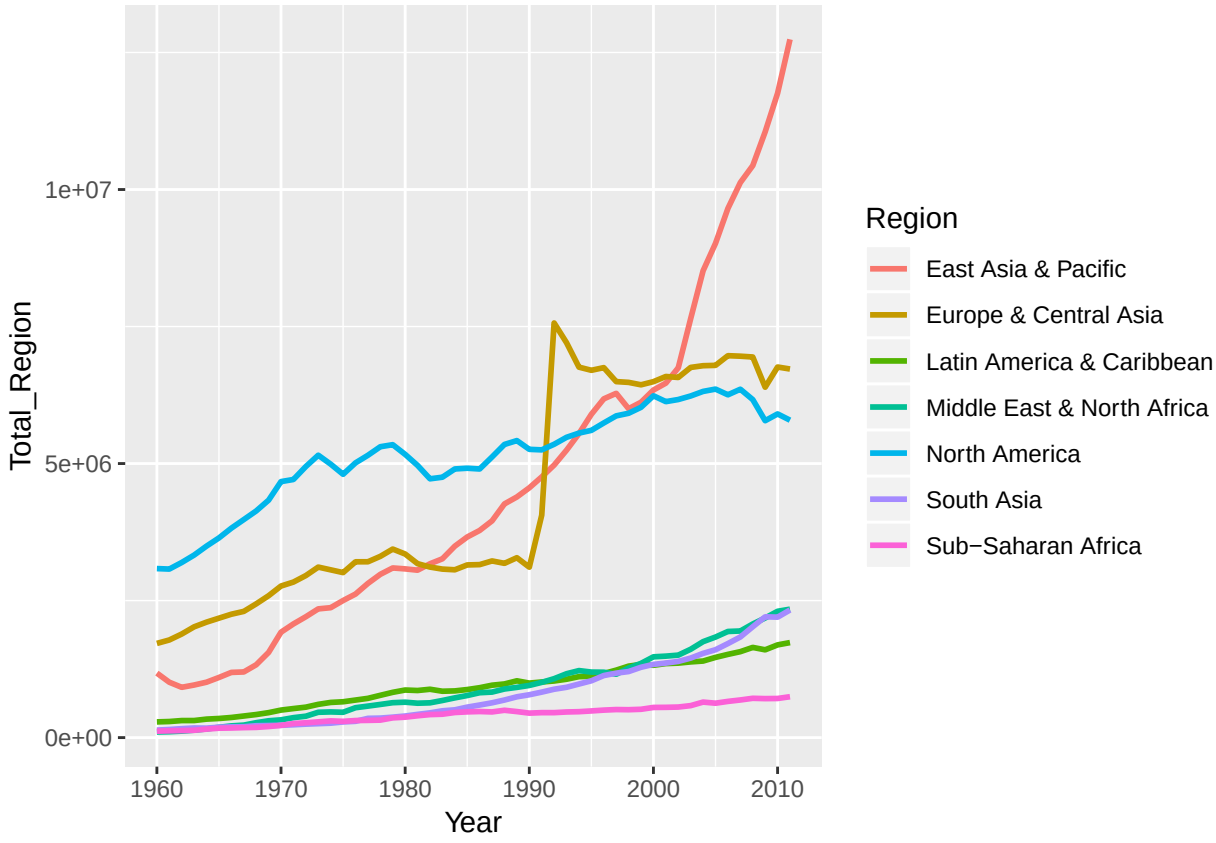
6.2.1 Contexto mundial en emisiones de contaminantes

Os pongo algunos enlaces. A partir de estos buscad más.

- Protocolo de kioto (wikipedia)
- Cambio Climático .org
- Acuerdo de París
- Acuerdo de París Comisón Europea

6.3 Un gráfico

```
all_data_CO2$`CO2 (kt) Pivoted`%>% group_by(`Year`,`Region`) %>% summarise(Total_Region=sum(`CO2 (kt)`,r
```



6.4 Conversiones desde los raw data y ajuste de metadatos

Primero por unas cuestiones de comodidad modificarermos los nombres de las tibbles y de las variables de cada tibble para no necesitar ponerlos entre “.

```
data_CO2=all_data_CO2
names(data_CO2)
```

```
## [1] "About"                    "CO2 (kt) Pivoted"
## [3] "CO2 (kt) RAW DATA"        "CO2 Data Cleaned"
## [5] "CO2 (kt) for Split"       "CO2 for World to Union"
## [7] "CO2 Per Capita RAW DATA"  "CO2 Per Capita (Pivoted)"
## [9] "Metadata - Countries"
```

```
long_names=names(data_CO2)# los guardo para el jefe
long_names
```

```
## [1] "About"                    "CO2 (kt) Pivoted"
## [3] "CO2 (kt) RAW DATA"        "CO2 Data Cleaned"
## [5] "CO2 (kt) for Split"       "CO2 for World to Union"
## [7] "CO2 Per Capita RAW DATA"  "CO2 Per Capita (Pivoted)"
## [9] "Metadata - Countries"
```

```
arreglo_names=function(x) return(gsub("_$","",gsub("(__+)","_",gsub("\\(|\\)|\\-|\\s","_",names(x)))))
names(data_CO2)=arreglo_names(data_CO2)
names(data_CO2)
```

```
## [1] "About"                   "CO2_kt_Pivoted"
## [3] "CO2_kt_RAW_DATA"         "CO2_Data_Cleaned"
## [5] "CO2_kt_for_Split"        "CO2_for_World_to_Union"
## [7] "CO2_Per_Capita_RAW_DATA" "CO2_Per_Capita_Pivoted"
## [9] "Metadata_Countries"
```

```
for(sheet in 1:length(data_CO2)) {
names(data_CO2[[sheet]])=arreglo_names(data_CO2[[sheet]])
}


print(data_CO2$CO2_kt_RAW_DATA,n=20,width = Inf)
```

```
## # A tibble: 248 x 60
##    Country_Name         Country_Code Indicator_Name     Indicator_Code
##    <chr>                <chr>        <chr>              <chr>
##  1 Aruba                ABW          CO2 emissions (kt) EN.ATM.CO2E.KT
##  2 Andorra              AND          CO2 emissions (kt) EN.ATM.CO2E.KT
##  3 Afghanistan          AFG          CO2 emissions (kt) EN.ATM.CO2E.KT
##  4 Angola               AGO          CO2 emissions (kt) EN.ATM.CO2E.KT
##  5 Albania              ALB          CO2 emissions (kt) EN.ATM.CO2E.KT
##  6 Arab World           ARB          CO2 emissions (kt) EN.ATM.CO2E.KT
##  7 United Arab Emirates ARE          CO2 emissions (kt) EN.ATM.CO2E.KT
##  8 Argentina            ARG          CO2 emissions (kt) EN.ATM.CO2E.KT
##  9 Armenia              ARM          CO2 emissions (kt) EN.ATM.CO2E.KT
## 10 American Samoa       ASM          CO2 emissions (kt) EN.ATM.CO2E.KT
## 11 Antigua and Barbuda  ATG          CO2 emissions (kt) EN.ATM.CO2E.KT
## 12 Australia            AUS          CO2 emissions (kt) EN.ATM.CO2E.KT
## 13 Austria              AUT          CO2 emissions (kt) EN.ATM.CO2E.KT
## 14 Azerbaijan           AZE          CO2 emissions (kt) EN.ATM.CO2E.KT
## 15 Burundi              BDI          CO2 emissions (kt) EN.ATM.CO2E.KT
## 16 Belgium              BEL          CO2 emissions (kt) EN.ATM.CO2E.KT
## 17 Benin                BEN          CO2 emissions (kt) EN.ATM.CO2E.KT
## 18 Burkina Faso         BFA          CO2 emissions (kt) EN.ATM.CO2E.KT
## 19 Bangladesh           BGD          CO2 emissions (kt) EN.ATM.CO2E.KT
## 20 Bulgaria             BGR          CO2 emissions (kt) EN.ATM.CO2E.KT
##    `1960`             `1961`             `1962`
##    <chr>              <chr>              <chr>
##  1 null               null               null
##  2 null               null               null
##  3 414.37099999999998 491.37799999999999 689.39599999999996
##  4 550.04999999999995 454.70800000000003 1180.7739999999999
##  5 2024.184           2280.8739999999998 2464.2240000000002
##  6 59563.989216993461 65151.095810624916 74357.70772706889
##  7 11.000999999999999 11.000999999999999 18.335000000000001
```

```
##  8 48815.103999999999 51180.319000000003 53695.881000000001
##  9 null               null               null
## 10 null               null               null
## 11 36.67              47.670999999999999 102.676
## 12 88202.350999999995 90589.567999999999 94912.960999999996
## 13 30821.134999999998 31862.562999999998 33905.082000000002
## 14 null               null               null
## 15 null               null               44.003999999999998
## 16 91000.271999999997 92793.434999999998 98117.918999999994
## 17 161.34800000000001 128.345            135.679
## 18 44.003999999999998 91.674999999999997 84.340999999999994
## 19 null               null               null
## 20 22295.360000000001 25973.361000000001 30736.794000000002
##    `1963`             `1964`             `1965`
##    <chr>              <chr>              <chr>
##  1 null               null               null
##  2 null               null               null
##  3 707.73099999999999 839.74300000000005 1008.425
##  4 1151.4380000000001 1224.778           1188.1079999999999
##  5 2082.8560000000002 2016.85            2174.5309999999999
##  6 87895.979157629205 103196.28157698263 123828.19687999594
##  7 22.001999999999999 18.335000000000001 22.001999999999999
##  8 50083.885999999999 55727.398999999998 58866.351000000002
##  9 null               null               null
## 10 null               null               null
## 11 84.340999999999994 91.674999999999997 150.34700000000001
## 12 101029.51700000001 108979.573         120966.996
## 13 36992.696000000004 38943.54           38188.137999999999
## 14 null               null               null
## 15 47.670999999999999 47.670999999999999 36.67
## 16 105781.94899999999 103662.423         105440.91800000001
## 17 121.011            143.01300000000001 150.34700000000001
## 18 88.007999999999996 110.01             102.676
## 19 null               null               null
## 20 34411.127999999997 42863.563000000002 46317.877
##    `1966`             `1967`             `1968`
##    <chr>              <chr>              <chr>
##  1 null               null               null
##  2 null               null               null
##  3 1092.7660000000001 1283.45            1224.778
##  4 1554.808           993.75699999999995 1672.152
##  5 2552.232           2680.5770000000002 3072.9459999999999
##  6 139079.15010611591 149730.95907415441 178805.03861785983
##  7 25.669             916.75             1243.1130000000001
##  8 63138.406000000003 65543.957999999999 69082.612999999998
##  9 null               null               null
## 10 null               null               null
## 11 348.36500000000001 564.71799999999996 990.09
## 12 120332.605         129265.417         134622.90400000001
## 13 39258.902000000002 39966.633000000002 42350.182999999997
## 14 null               null               null
## 15 47.670999999999999 47.670999999999999 55.005000000000003
## 16 105206.23          107472.436         118557.777
## 17 113.67700000000001 143.01300000000001 154.01400000000001
```

```
## 18 102.676            102.676            102.676
## 19 null               null               null
## 20 48767.432999999997 55166.347999999998 59526.411
##    `1969`             `1970`             `1971`
##    <chr>              <chr>              <chr>
##  1 null               null               null
##  2 null               null               null
##  3 942.41899999999998 1672.152           1895.8389999999999
##  4 2786.92            3582.6590000000001 3410.31
##  5 3245.2950000000001 3744.0070000000001 4352.7290000000003
##  6 213131.61252175251 220575.39907832802 250915.80325813728
##  7 20524.199000000001 15247.386          21184.258999999998
##  8 77329.695999999996 82734.854000000007 88939.418000000005
##  9 null               null               null
## 10 null               null               null
## 11 1257.7809999999999 462.04199999999997 425.37200000000001
## 12 142257.598         147618.75200000001 152774.554
## 13 44693.396000000001 50692.608          52155.741000000002
## 14 null               null               null
## 15 73.34              62.338999999999999 73.34
## 16 123618.23699999999 125620.41899999999 121528.04700000001
## 17 201.685            282.35899999999998 293.36
## 18 121.011            143.01300000000001 150.34700000000001
## 19 null               null               null
## 20 66376.366999999998 61238.9            64300.845000000001
##    `1972`             `1973`             `1974`
##    <chr>              <chr>              <chr>
##  1 null               null               null
##  2 null               null               null
##  3 1532.806           1639.1489999999999 1917.8409999999999
##  4 4506.7430000000004 4880.777           4873.4430000000002
##  5 5643.5129999999999 5291.4809999999998 4345.3950000000004
##  6 272872.77129632013 318599.93743133865 310921.23847850045
##  7 23454.132000000001 30630.451000000001 31327.181
##  8 90156.861999999994 94065.884000000005 95569.354000000007
##  9 null               null               null
## 10 null               null               null
## 11 374.03399999999999 330.03             429.03899999999999
## 12 157486.649         170992.21          172356.334
## 13 56193.108          60164.468999999997 57469.224000000002
## 14 null               null               null
## 15 73.34              73.34              91.674999999999997
## 16 130834.893         138872.95699999999 135220.625
## 17 388.702            381.36799999999999 407.03699999999998
## 18 161.34800000000001 168.68199999999999 205.352
## 19 3509.319           4554.4139999999998 4660.7569999999996
## 20 66171.014999999999 68825.922999999995 71260.811000000002
##    `1975`             `1976`             `1977`
##    <chr>              <chr>              <chr>
##  1 null               null               null
##  2 null               null               null
##  3 2126.86            1987.5139999999999 2390.884
##  4 4415.0680000000002 3285.6320000000001 3534.9879999999998
##  5 4594.7510000000002 4950.45            5720.52
```

```
##  6 307786.62619086955 373496.0280329872  395284.15278635628
##  7 31070.491000000002 39651.271000000001 38785.858999999997
##  8 94931.296000000002 99786.403999999995 100791.162
##  9 null               null               null
## 10 null               null               null
## 11 707.73099999999999 403.37             465.709
## 12 175883.98800000001 174244.83900000001 187787.07
## 13 54392.610999999997 58415.31           56218.777000000002
## 14 null               null               null
## 15 77.007000000000005 88.007999999999996 99.009
## 16 122100.099         130009.818         126570.17200000001
## 17 443.70699999999999 260.35700000000003 297.02699999999999
## 18 220.02             209.01900000000001 249.35599999999999
## 19 4869.7759999999998 5570.1729999999998 5812.1949999999997
## 20 73061.308000000005 73116.312999999995 75969.239000000001
##    `1978`             `1979`             `1980`
##    <chr>              <chr>              <chr>
##  1 null               null               null
##  2 null               null               null
##  3 2159.8629999999998 2240.5369999999998 1760.16
##  4 5412.4920000000002 5504.1670000000004 5346.4859999999999
##  5 6494.2569999999996 7587.0230000000001 5170.47
##  6 425910.85644911078 456159.49797412992 508779.87473698321
##  7 44814.406999999999 36607.661          36904.688000000002
##  8 102639.33          110703.06299999999 108737.55100000001
##  9 null               null               null
## 10 null               null               null
## 11 491.37799999999999 407.03699999999998 143.01300000000001
## 12 202015.03          205069.641         220746.06599999999
## 13 57483.892          61594.599000000002 52306.088000000003
## 14 null               null               null
## 15 102.676            110.01             146.68
## 16 135855.016         140233.41399999999 135301.299
## 17 363.03300000000002 366.7              517.04700000000003
## 18 348.36500000000001 407.03699999999998 432.70600000000002
## 19 6017.5469999999996 6648.2709999999997 7638.3609999999999
## 20 81411.066999999995 79152.195000000007 77487.376999999993
##    `1981`             `1982`             `1983`
##    <chr>              <chr>              <chr>
##  1 null               null               null
##  2 null               null               null
##  3 1983.847           2101.1909999999998 2522.8960000000002
##  4 5280.48            4649.7560000000003 5115.4650000000001
##  5 7341.3339999999998 7308.3310000000001 7631.027
##  6 497496.73870398867 476949.24504338234 508651.40702027711
##  7 36857.017          36871.684999999998 35342.546000000002
##  8 102041.609         103424.068         105213.564
##  9 null               null               null
## 10 null               null               null
## 11 106.343            293.36             84.340999999999994
## 12 230360.94          234119.61499999999 225003.45300000001
## 13 56130.769          53868.23           51983.392
## 14 null               null               null
## 15 157.68100000000001 157.68100000000001 205.352
```

```
## 16 124021.607         117553.019         101630.905
## 17 429.03899999999999 502.37900000000002 462.04199999999997
## 18 557.38400000000001 575.71900000000005 594.05399999999997
## 19 7931.7209999999995 8599.1149999999998 8236.0820000000003
## 20 80343.97           90131.192999999999 90365.880999999994
##    `1984`             `1985`             `1986`
##    <chr>              <chr>              <chr>
##  1 null               null               179.68299999999999
##  2 null               null               null
##  3 2830.924           3509.319           3142.6190000000001
##  4 5009.1220000000003 4701.0940000000001 4660.7569999999996
##  5 7825.3779999999997 7880.3829999999998 8056.3990000000003
##  6 554903.45554069546 587093.79484107369 648475.66988328891
##  7 46394.883999999998 49926.205000000002 47234.627
##  8 106522.683         100596.811         104212.473
##  9 null               null               null
## 10 null               null               null
## 11 146.68             249.35599999999999 249.35599999999999
## 12 236594.84          241229.92800000001 239964.81299999999
## 13 54550.292000000001 54700.639000000003 54080.915999999997
## 14 null               null               null
## 15 220.02             231.02099999999999 242.02199999999999
## 16 105422.583         104472.83          102874.018
## 17 513.38             744.40099999999995 693.06299999999999
## 18 465.709            476.71             480.37700000000001
## 19 9123.4959999999992 10234.597          11463.041999999999
## 20 87366.274999999994 89540.805999999997 91528.320000000007
##    `1987`             `1988`             `1989`
##    <chr>              <chr>              <chr>
##  1 447.37400000000002 612.38900000000001 649.05899999999997
##  2 null               null               null
##  3 3124.2840000000001 2867.5940000000001 2775.9189999999999
##  4 5815.8620000000001 5130.1329999999998 5009.1220000000003
##  5 7444.01            7326.6660000000002 8984.15
##  6 649749.33553177584 688649.3601504087  699737.95935524825
##  7 47693.002          48367.73           54487.953000000001
##  8 114942.11500000001 121473.042         117090.977
##  9 null               null               null
## 10 null               null               null
## 11 275.02499999999998 286.02600000000001 286.02600000000001
## 12 256106.94699999999 261145.405         277771.58299999998
## 13 57744.249000000003 53340.182000000001 54117.586000000003
## 14 null               null               null
## 15 267.69099999999997 256.69             300.69400000000002
## 16 103116.04          100354.789         107461.435
## 17 539.04899999999998 561.05100000000004 641.72500000000002
## 18 517.04700000000003 553.71699999999998 821.40800000000002
## 19 11862.745000000001 13545.897999999999 13454.223
## 20 91634.663          87289.267999999996 86739.217999999993
##    `1990`             `1991`             `1992`
##    <chr>              <chr>              <chr>
##  1 1840.8340000000001 1928.8420000000001 1723.49
##  2 null               null               null
##  3 2676.91            2493.56            1426.463
```

```
##  4 4429.7359999999999 4367.3969999999999 4418.7349999999997
##  5 7488.0140000000001 3971.3609999999999 2387.2170000000001
##  6 711039.44791920087 749355.86205336847 809684.30181859317
##  7 52009.061000000002 57010.849000000002 58136.618000000002
##  8 112613.57          117021.304         121447.37300000001
##  9 null               null               4052.0349999999999
## 10 null               null               null
## 11 300.69400000000002 289.69299999999998 289.69299999999998
## 12 263847.984         261574.44399999999 268068.701
## 13 57722.247000000003 61638.603000000003 56688.152999999998
## 14 null               null               57678.243000000002
## 15 293.36             337.36399999999998 308.02800000000002
## 16 106049.64          110959.753         111968.178
## 17 715.06500000000005 828.74199999999996 905.74900000000002
## 18 586.72             627.05700000000002 630.72400000000005
## 19 15533.412          15940.449000000001 17748.28
## 20 75763.887000000002 58176.955000000002 54458.616999999998
##    `1993`             `1994`             `1995`
##    <chr>              <chr>              <chr>
##  1 1771.1610000000001 1763.827           1782.162
##  2 null               null               407.03699999999998
##  3 1375.125           1320.12            1268.7819999999999
##  4 5801.1940000000004 3890.6869999999999 11012.001
##  5 2343.2130000000002 1928.8420000000001 2086.5230000000001
##  6 886394.21071695175 914334.10384745116 876887.59968069405
##  7 65980.331000000006 73130.981          70641.088000000003
##  8 118609.11500000001 123350.546         122547.473
##  9 2896.93            2966.6030000000001 3490.9839999999999
## 10 null               null               null
## 11 304.36099999999999 311.69499999999999 322.69600000000003
## 12 277478.223         278204.28899999999 281940.962
## 13 57135.527000000002 57095.19           59827.105000000003
## 14 49365.154000000002 42672.879000000001 33479.71
## 15 326.363            333.697            322.69600000000003
## 16 107652.11900000001 112334.878         112327.54399999999
## 17 1133.1030000000001 1265.115           1327.454
## 18 627.05700000000002 645.39200000000005 627.05700000000002
## 19 17407.249          18969.391          22816.074000000001
## 20 68077.854999999996 54344.94           58004.606
##    `1996`             `1997`             `1998`
##    <chr>              <chr>              <chr>
##  1 1800.4970000000001 1837.1669999999999 1712.489
##  2 425.37200000000001 458.375            484.04399999999998
##  3 1199.1089999999999 1114.768           1056.096
##  4 10491.287          7381.6710000000003 7308.3310000000001
##  5 2016.85            1543.807           1752.826
##  6 869025.27668713185 836526.20675949391 906453.7954654101
##  7 41059.398999999998 41646.118999999999 81495.407999999996
##  8 129217.746         134677.90900000001 137673.848
##  9 2607.2370000000001 3278.2979999999998 3406.643
## 10 null               null               null
## 11 322.69600000000003 337.36399999999998 333.697
## 12 302241.47399999999 305838.80099999998 316964.47899999999
## 13 63226.413999999997 62705.7            63717.792000000001
```

```
## 14 31510.530999999999 29809.043000000001 31675.545999999998
## 15 319.029            304.36099999999999 293.36
## 16 118059.065         115341.818         118374.427
## 17 1265.115           1217.444           1213.777
## 18 707.73099999999999 806.74             861.745
## 19 24029.850999999999 25063.945          24048.186000000002
## 20 56559.807999999997 51616.692000000003 48998.453999999998
##    `1999`             `2000`             `2001`
##    <chr>              <chr>              <chr>
##  1 1749.1590000000001 2321.2109999999998 2357.8809999999999
##  2 513.38             524.38099999999997 524.38099999999997
##  3 832.40899999999999 781.07100000000003 645.39200000000005
##  4 9156.4989999999998 9541.5339999999997 9732.2180000000008
##  5 2984.9380000000001 3021.6080000000002 3223.2930000000001
##  6 920061.42539774161 1045912.7410000002 1029730.27
##  7 78374.790999999997 112562.232         101414.552
##  8 145488.22500000001 141076.82399999999 132631.723
##  9 3058.2779999999998 3465.3150000000001 3542.3220000000001
## 10 null               null               null
## 11 348.36500000000001 344.69799999999998 344.69799999999998
## 12 325380.24400000001 329479.95          324877.86499999999
## 13 61917.294999999998 62188.652999999998 65785.98
## 14 28576.931          29508.348999999998 28771.281999999999
## 15 286.02600000000001 289.69299999999998 205.352
## 16 115085.128         114894.444         114315.058
## 17 1562.1420000000001 1598.8119999999999 1815.165
## 18 931.41800000000001 1041.4280000000001 997.42399999999998
## 19 25236.294000000002 27869.200000000001 32456.616999999998
## 20 43981.998          43530.957000000002 46453.555999999997
##    `2002`             `2003`             `2004`
##    <chr>              <chr>              <chr>
##  1 2372.549           2416.5529999999999 2420.2199999999998
##  2 531.71500000000003 535.38199999999995 564.71799999999996
##  3 894.74800000000005 1037.761           957.08699999999999
##  4 12665.817999999999 9064.8240000000005 18793.375
##  5 3751.3409999999999 4294.0569999999998 4165.7120000000004
##  6 1050712.844        1138783.183        1254917.0730000001
##  7 84704.032999999996 106841.712         113240.62699999999
##  8 123266.205         133126.76800000001 156170.196
##  9 3043.61            3428.645           3644.998
## 10 null               null               null
## 11 363.03300000000002 388.702            407.03699999999998
## 12 341390.36599999998 336355.57500000001 342776.49200000003
## 13 66977.755000000005 72074.884999999995 72272.903000000006
## 14 29614.691999999999 30615.782999999999 32089.917000000001
## 15 212.68600000000001 161.34800000000001 198.018
## 16 107098.402         114564.414         110908.41499999999
## 17 2079.1889999999999 2354.2139999999999 2508.2280000000001
## 18 1004.758           1078.098           1103.7670000000001
## 19 33707.063999999998 33883.08           39750.28
## 20 44634.724000000002 47307.966999999997 46787.252999999997
##    `2005`             `2006`             `2007`
##    <chr>              <chr>              <chr>
##  1 2497.2269999999999 2497.2269999999999 2592.569
```

```
##  2 575.71900000000005 546.38300000000004 539.04899999999998
##  3 1338.4549999999999 1657.4839999999999 2280.8739999999998
##  4 19156.407999999999 22266.024000000001 25151.953000000001
##  5 4253.72            3865.018           4477.4070000000002
##  6 1318865.8859999997 1375301.0159999998 1374211.9169999999
##  7 116148.558         123874.927         139404.67199999999
##  8 160951.96400000001 174237.505         179738.005
##  9 4352.7290000000003 4382.0649999999996 5064.1270000000004
## 10 null               null               null
## 11 410.70400000000001 425.37200000000001 469.37599999999998
## 12 350268.17300000001 357946.87099999998 363795.73599999998
## 13 74278.751999999993 72214.231          69724.338000000003
## 14 34337.788          39167.226999999999 30509.439999999999
## 15 154.01400000000001 187.017            187.017
## 16 108297.511         106775.70600000001 103215.049
## 17 2394.5509999999999 3872.3519999999999 4495.7420000000002
## 18 1125.769           1360.4570000000001 1646.4829999999999
## 19 37553.747000000003 48136.709000000003 46886.262000000002
## 20 47909.355000000003 48943.449000000001 52188.743999999999
##    `2008`             `2009`             `2010`
##    <chr>              <chr>              <chr>
##  1 2508.2280000000001 2522.8960000000002 2456.89
##  2 539.04899999999998 517.04700000000003 517.04700000000003
##  3 4217.05            6776.616           8470.77
##  4 27172.47           29361.669000000002 29743.037
##  5 4657.09            4488.4080000000004 4415.0680000000002
##  6 1472417.844        1577099.6929999997 1680300.0740000003
##  7 158935.114         162602.114         167596.568
##  8 191578.74799999999 180654.755         179000.93799999999
##  9 5559.1719999999996 4360.0630000000001 4217.05
## 10 null               null               null
## 11 480.37700000000001 509.71300000000002 524.38099999999997
## 12 376197.53          382464.43300000002 368170.467
## 13 68752.582999999999 63483.103999999999 67975.179000000004
## 14 35503.894          31902.9            30678.121999999999
## 15 190.684            190.684            194.351
## 16 103882.443         104021.789         109093.25
## 17 4488.4080000000004 4752.4319999999998 5152.1350000000002
## 18 1697.8209999999999 1664.818           1683.153
## 19 49581.506999999998 52790.131999999998 56152.771000000001
## 20 50791.616999999998 42654.544000000002 44128.678
##    `2011`             `2012` `2013` `2014` `2015`
##    <chr>              <lgl>  <lgl>  <lgl>  <lgl>
##  1 2438.5549999999998 NA     NA     NA     NA
##  2 491.37799999999999 NA     NA     NA     NA
##  3 12251.447          NA     NA     NA     NA
##  4 29710.034          NA     NA     NA     NA
##  5 4668.0910000000003 NA     NA     NA     NA
##  6 1704417.9330000002 NA     NA     NA     NA
##  7 178483.891         NA     NA     NA     NA
##  8 190034.94099999999 NA     NA     NA     NA
##  9 4961.451           NA     NA     NA     NA
## 10 null               NA     NA     NA     NA
## 11 513.38             NA     NA     NA     NA
```

```
## 12 369039.54599999997 NA     NA     NA     NA
## 13 65202.927000000003 NA     NA     NA     NA
## 14 33457.707999999999 NA     NA     NA     NA
## 15 209.01900000000001 NA     NA     NA     NA
## 16 97765.887000000002 NA     NA     NA     NA
## 17 4987.12            NA     NA     NA     NA
## 18 1932.509           NA     NA     NA     NA
## 19 57069.521000000001 NA     NA     NA     NA
## 20 49339.485000000001 NA     NA     NA     NA
## # ... with 228 more rows
```

```
glimpse(data_CO2$CO2_kt_RAW_DATA) # es similar en algunos caso a str pero es mas adecuado para tibbñles
```

```
## Observations: 248
## Variables: 60
## $ Country_Name   <chr> "Aruba", "Andorra", "Afghanistan", "Angola", "A...
## $ Country_Code   <chr> "ABW", "AND", "AFG", "AGO", "ALB", "ARB", "ARE"...
## $ Indicator_Name <chr> "CO2 emissions (kt)", "CO2 emissions (kt)", "CO...
## $ Indicator_Code <chr> "EN.ATM.CO2E.KT", "EN.ATM.CO2E.KT", "EN.ATM.CO2...
## $ `1960`         <chr> "null", "null", "414.37099999999998", "550.0499...
## $ `1961`         <chr> "null", "null", "491.37799999999999", "454.7080...
## $ `1962`         <chr> "null", "null", "689.39599999999996", "1180.773...
## $ `1963`         <chr> "null", "null", "707.73099999999999", "1151.438...
## $ `1964`         <chr> "null", "null", "839.74300000000005", "1224.778...
## $ `1965`         <chr> "null", "null", "1008.425", "1188.1079999999999...
## $ `1966`         <chr> "null", "null", "1092.7660000000001", "1554.808...
## $ `1967`         <chr> "null", "null", "1283.45", "993.75699999999995"...
## $ `1968`         <chr> "null", "null", "1224.778", "1672.152", "3072.9...
## $ `1969`         <chr> "null", "null", "942.41899999999998", "2786.92"...
## $ `1970`         <chr> "null", "null", "1672.152", "3582.6590000000001...
## $ `1971`         <chr> "null", "null", "1895.8389999999999", "3410.31"...
## $ `1972`         <chr> "null", "null", "1532.806", "4506.7430000000004...
## $ `1973`         <chr> "null", "null", "1639.1489999999999", "4880.777...
## $ `1974`         <chr> "null", "null", "1917.8409999999999", "4873.443...
## $ `1975`         <chr> "null", "null", "2126.86", "4415.0680000000002"...
## $ `1976`         <chr> "null", "null", "1987.5139999999999", "3285.632...
## $ `1977`         <chr> "null", "null", "2390.884", "3534.9879999999998...
## $ `1978`         <chr> "null", "null", "2159.8629999999998", "5412.492...
## $ `1979`         <chr> "null", "null", "2240.5369999999998", "5504.167...
## $ `1980`         <chr> "null", "null", "1760.16", "5346.4859999999999"...
## $ `1981`         <chr> "null", "null", "1983.847", "5280.48", "7341.33...
## $ `1982`         <chr> "null", "null", "2101.1909999999998", "4649.756...
## $ `1983`         <chr> "null", "null", "2522.8960000000002", "5115.465...
## $ `1984`         <chr> "null", "null", "2830.924", "5009.1220000000003...
## $ `1985`         <chr> "null", "null", "3509.319", "4701.0940000000001...
## $ `1986`         <chr> "179.68299999999999", "null", "3142.61900000000...
## $ `1987`         <chr> "447.37400000000002", "null", "3124.28400000000...
## $ `1988`         <chr> "612.38900000000001", "null", "2867.59400000000...
## $ `1989`         <chr> "649.05899999999997", "null", "2775.91899999999...
## $ `1990`         <chr> "1840.8340000000001", "null", "2676.91", "4429.....
## $ `1991`         <chr> "1928.8420000000001", "null", "2493.56", "4367.....
## $ `1992`         <chr> "1723.49", "null", "1426.463", "4418.7349999999...
## $ `1993`         <chr> "1771.1610000000001", "null", "1375.125", "5801...
## $ `1994`         <chr> "1763.827", "null", "1320.12", "3890.6869999999...
## $ `1995`         <chr> "1782.162", "407.03699999999998", "1268.7819999...
```

```
## $ `1996`         <chr> "1800.4970000000001", "425.37200000000001", "11...
## $ `1997`         <chr> "1837.1669999999999", "458.375", "1114.768", "7...
## $ `1998`         <chr> "1712.489", "484.04399999999998", "1056.096", "...
## $ `1999`         <chr> "1749.1590000000001", "513.38", "832.4089999999...
## $ `2000`         <chr> "2321.2109999999998", "524.38099999999997", "78...
## $ `2001`         <chr> "2357.8809999999999", "524.38099999999997", "64...
## $ `2002`         <chr> "2372.549", "531.71500000000003", "894.74800000...
## $ `2003`         <chr> "2416.5529999999999", "535.38199999999995", "10...
## $ `2004`         <chr> "2420.2199999999998", "564.71799999999996", "95...
## $ `2005`         <chr> "2497.2269999999999", "575.71900000000005", "13...
## $ `2006`         <chr> "2497.2269999999999", "546.38300000000004", "16...
## $ `2007`         <chr> "2592.569", "539.04899999999998", "2280.8739999...
## $ `2008`         <chr> "2508.2280000000001", "539.04899999999998", "42...
## $ `2009`         <chr> "2522.8960000000002", "517.04700000000003", "67...
## $ `2010`         <chr> "2456.89", "517.04700000000003", "8470.77", "29...
## $ `2011`         <chr> "2438.5549999999998", "491.37799999999999", "12...
## $ `2012`         <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ `2013`         <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ `2014`         <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ `2015`         <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
```

```
#str(all_data_CO2$`CO2_(kt)_RAW_DATA`)

library(naniar)# cargar para la función replace_with_na. El paquete naniar aporta utilizades para elimi
# ver un manual en https://cran.r-project.org/web/packages/naniar/vignettes/getting-started-w-naniar.ht
data_CO2$CO2_kt_RAW_DATA %>% gather(`1960`:`2015`, key="Year", value="CO2") %>%
  naniar::replace_with_na(replace =list(CO2 ="null")) %>%
  mutate(Year=as.integer(Year),CO2=as.numeric(CO2)) %>%
  arrange(Country_Code) -> aux


print(aux,n=10,width=Inf)
```

```
## # A tibble: 13,888 x 6
##    Country_Name Country_Code Indicator_Name     Indicator_Code  Year   CO2
##    <chr>        <chr>        <chr>              <chr>          <int> <dbl>
##  1 Aruba        ABW          CO2 emissions (kt) EN.ATM.CO2E.KT  1960    NA
##  2 Aruba        ABW          CO2 emissions (kt) EN.ATM.CO2E.KT  1961    NA
##  3 Aruba        ABW          CO2 emissions (kt) EN.ATM.CO2E.KT  1962    NA
##  4 Aruba        ABW          CO2 emissions (kt) EN.ATM.CO2E.KT  1963    NA
##  5 Aruba        ABW          CO2 emissions (kt) EN.ATM.CO2E.KT  1964    NA
##  6 Aruba        ABW          CO2 emissions (kt) EN.ATM.CO2E.KT  1965    NA
##  7 Aruba        ABW          CO2 emissions (kt) EN.ATM.CO2E.KT  1966    NA
##  8 Aruba        ABW          CO2 emissions (kt) EN.ATM.CO2E.KT  1967    NA
##  9 Aruba        ABW          CO2 emissions (kt) EN.ATM.CO2E.KT  1968    NA
## 10 Aruba        ABW          CO2 emissions (kt) EN.ATM.CO2E.KT  1969    NA
## # ... with 1.388e+04 more rows
```

```
periodos=table(data_CO2$CO2_kt_Pivoted$Year)
periodos
```

```
##
## 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974
##  213  214  214  214  214  214  214  214  214  214  214  214  214  214  214
## 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989
```

```
##  214  214  214  214  214  214  214  214  214  214  214  214  214  214  214
## 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004
##  214  214  214  214  214  214  214  214  214  214  214  214  214  214  214
## 2005 2006 2007 2008 2009 2010 2011
##  214  214  214  214  214  214  214
```

```
all(periodos==214)# hay alguno el primero 1960 que parece que falta algo
```

```
## [1] FALSE
```

```
year_country=table(data_CO2$CO2_kt_Pivoted$Year,data_CO2$CO2_kt_Pivoted$Country_Code)
#year_country # es muy grande mejor contemos las frecuencias de apariciones
table(year_country)
```

```
## year_country
##     0     1
##     1 11127
```

```
str(year_country)
```

```
##  'table' int [1:52, 1:214] 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:52] "1960" "1961" "1962" "1963" ...
##   ..$ : chr [1:214] "ABW" "AFG" "AGO" "ALB" ...
```

```
indice=which(year_country==0,arr.ind = TRUE)
indice
```

```
##      row col
## 1960   1 174
```

```
dimnames(year_country)[[1]][indice[1]]
```

```
## [1] "1960"
```

```
dimnames(year_country)[[2]][indice[2]]
```

```
## [1] "SMR"
```

```
data_CO2$CO2_kt_Pivoted[data_CO2$CO2_kt_Pivoted$Country_Code=="SMR",]
```

```
## # A tibble: 51 x 5
##    Country_Name Country_Code Region                Year CO2_kt
##    <chr>        <chr>        <chr>                <dbl>  <dbl>
##  1 San Marino   SMR          Europe & Central Asia  1961     NA
##  2 San Marino   SMR          Europe & Central Asia  1962     NA
##  3 San Marino   SMR          Europe & Central Asia  1963     NA
##  4 San Marino   SMR          Europe & Central Asia  1964     NA
##  5 San Marino   SMR          Europe & Central Asia  1965     NA
##  6 San Marino   SMR          Europe & Central Asia  1966     NA
##  7 San Marino   SMR          Europe & Central Asia  1967     NA
##  8 San Marino   SMR          Europe & Central Asia  1968     NA
##  9 San Marino   SMR          Europe & Central Asia  1969     NA
## 10 San Marino   SMR          Europe & Central Asia  1970     NA
## # ... with 41 more rows
```

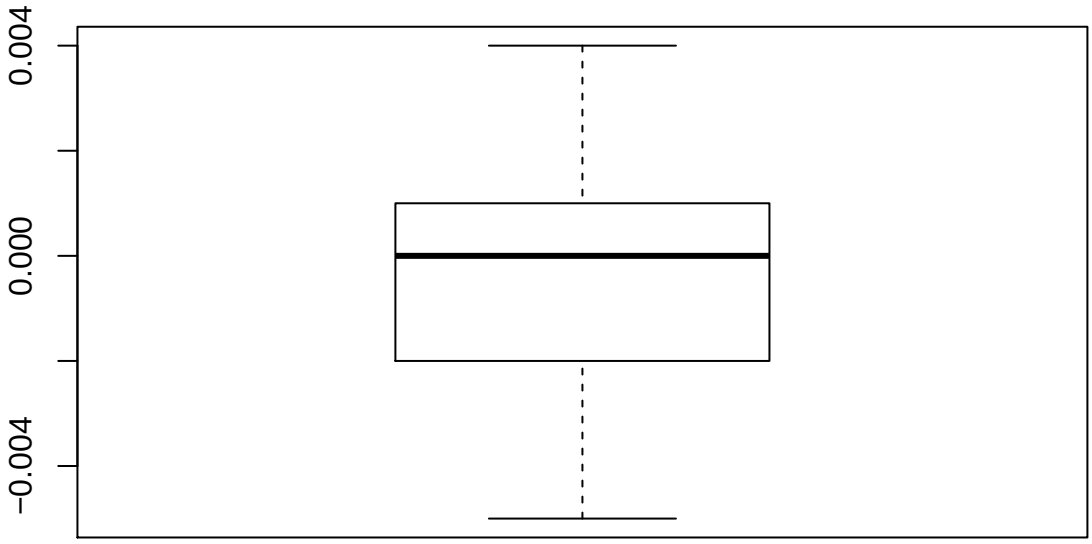
```
inner_join(aux,data_CO2$CO2_kt_Pivoted) %>% mutate(dif=CO2-CO2_kt) -> aux2
```

```
## Joining, by = c("Country_Name", "Country_Code", "Year")
```

```
summary(aux2$dif)# los errores pueden ser debidos al redondeo al convertir as.numeric(CO2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -5e-03  -2e-03   0e+00  -3e-04   1e-03   4e-03    2095
```

```
boxplot(aux2$dif)
```




```
print(aux2,whihd=Inf)
```

```
## # A tibble: 11,127 x 9
##    Country_Name Country_Code Indicator_Name Indicator_Code  Year   CO2
##    <chr>        <chr>        <chr>          <chr>          <dbl> <dbl>
##  1 Aruba        ABW          CO2 emissions~ EN.ATM.CO2E.KT  1960    NA
##  2 Aruba        ABW          CO2 emissions~ EN.ATM.CO2E.KT  1961    NA
##  3 Aruba        ABW          CO2 emissions~ EN.ATM.CO2E.KT  1962    NA
##  4 Aruba        ABW          CO2 emissions~ EN.ATM.CO2E.KT  1963    NA
##  5 Aruba        ABW          CO2 emissions~ EN.ATM.CO2E.KT  1964    NA
##  6 Aruba        ABW          CO2 emissions~ EN.ATM.CO2E.KT  1965    NA
##  7 Aruba        ABW          CO2 emissions~ EN.ATM.CO2E.KT  1966    NA
##  8 Aruba        ABW          CO2 emissions~ EN.ATM.CO2E.KT  1967    NA
##  9 Aruba        ABW          CO2 emissions~ EN.ATM.CO2E.KT  1968    NA
## 10 Aruba        ABW          CO2 emissions~ EN.ATM.CO2E.KT  1969    NA
## # ... with 11,117 more rows, and 3 more variables: Region <chr>,
## #   CO2_kt <dbl>, dif <dbl>
```

```
data_clean=aux2 %>% inner_join(data_CO2$CO2_Per_Capita_Pivoted) %>% inner_join(data_CO2$Metadata_Countr
```

```
## Joining, by = c("Country_Name", "Country_Code", "Year")
## Joining, by = c("Country_Code", "Region")
```

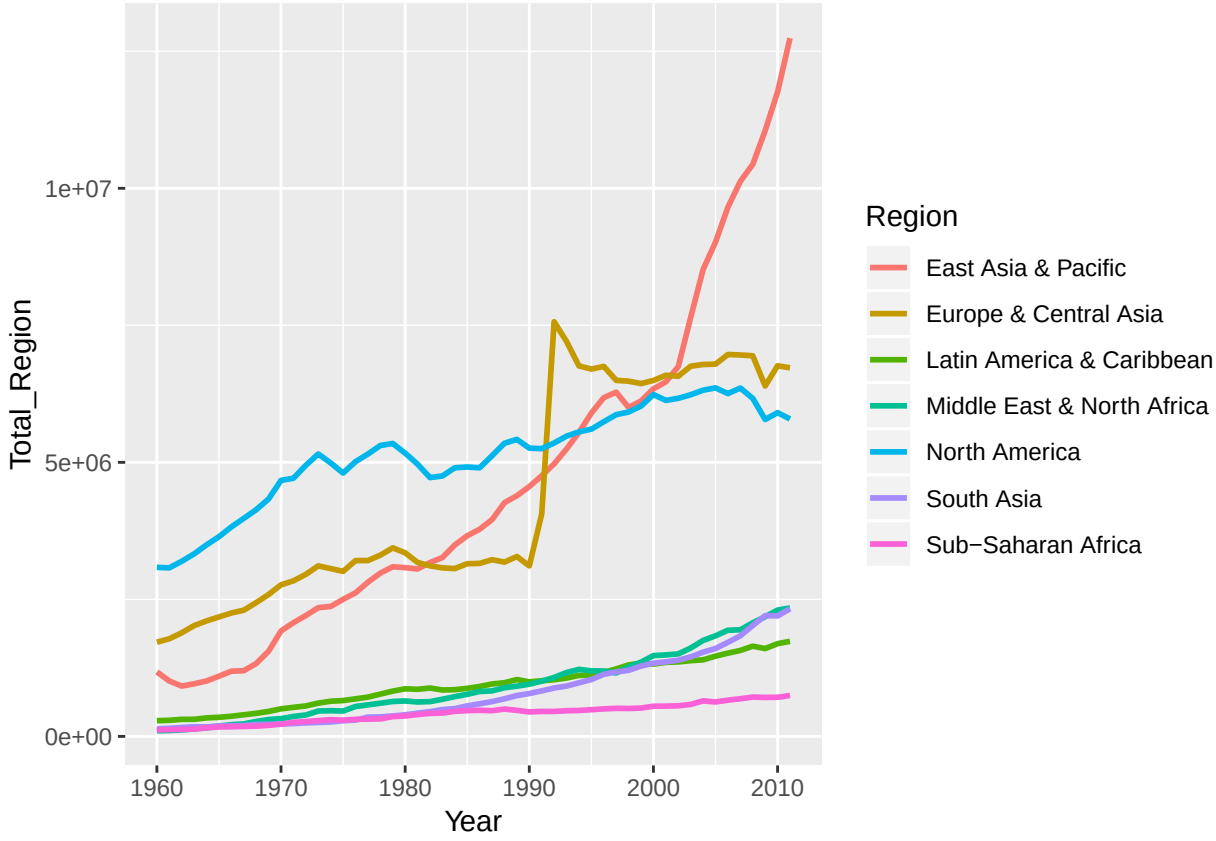
6.4.1 Preguntas y gráficos

```
glimpse(data_clean)
```

```
## Observations: 11,127
## Variables: 13
## $ Country_Name               <chr> "Aruba", "Aruba", "Aruba", "Aruba",...
## $ Country_Code               <chr> "ABW", "ABW", "ABW", "ABW", "ABW", ...
```

```
## $ Indicator_Name             <chr> "CO2 emissions (kt)", "CO2 emission...
## $ Indicator_Code             <chr> "EN.ATM.CO2E.KT", "EN.ATM.CO2E.KT",...
## $ Year                       <dbl> 1960, 1961, 1962, 1963, 1964, 1965,...
## $ CO2                        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ Region                     <chr> "Latin America & Caribbean", "Latin...
## $ CO2_kt                     <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ dif                        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ CO2_Per_Capita_metric_tons <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ IncomeGroup                <chr> "High income: nonOECD", "High incom...
## $ SpecialNotes               <chr> "SNA data for 2000-2011 are updated...
## $ TableName                  <chr> "Aruba", "Aruba", "Aruba", "Aruba",...
```

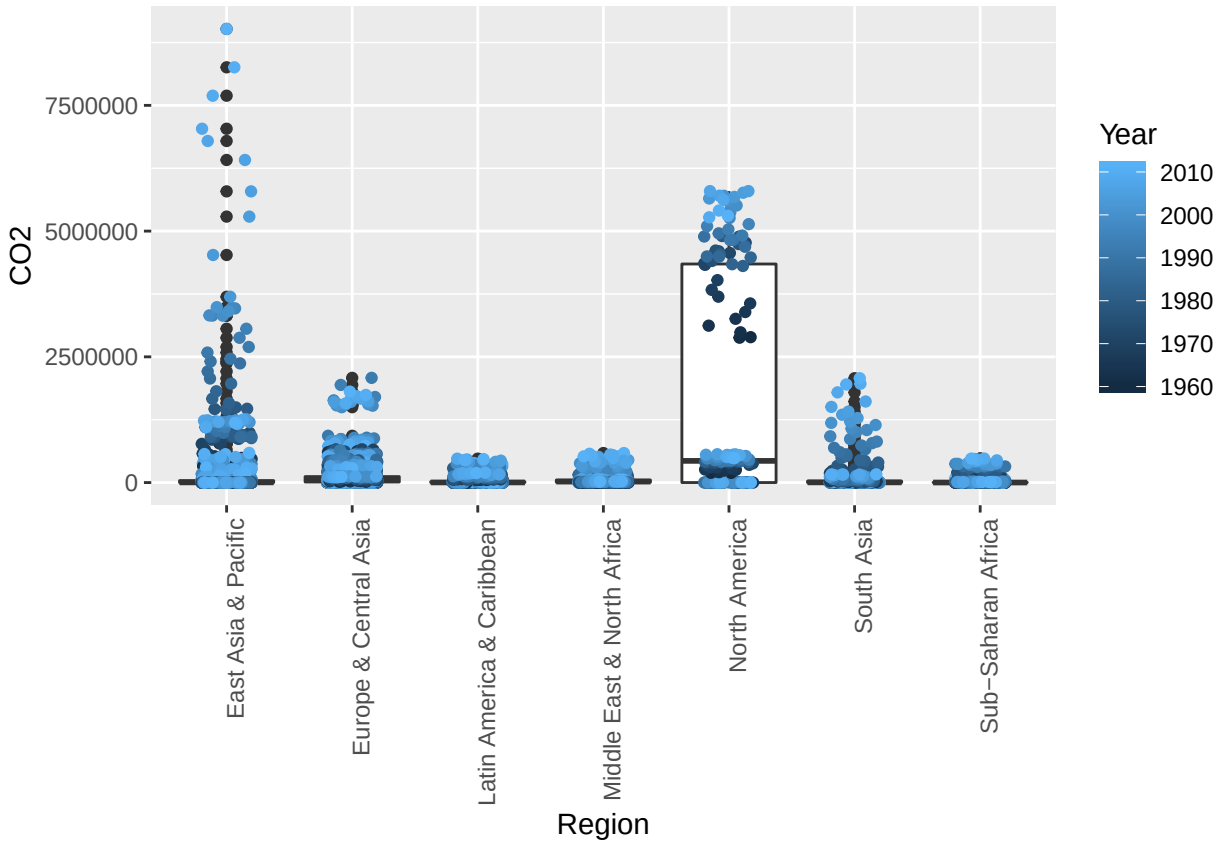
```
data_clean %>% group_by(Year,Region) %>% summarise(Total_Region=sum(CO2_kt,na.rm=TRUE)) %>% ggplot(aes(
```




```
data_clean %>% ggplot(aes(Region,CO2,colour=Year)) + geom_boxplot() + theme(axis.text.x = element_text(a
```

```
## Warning: Removed 2095 rows containing non-finite values (stat_boxplot).
```

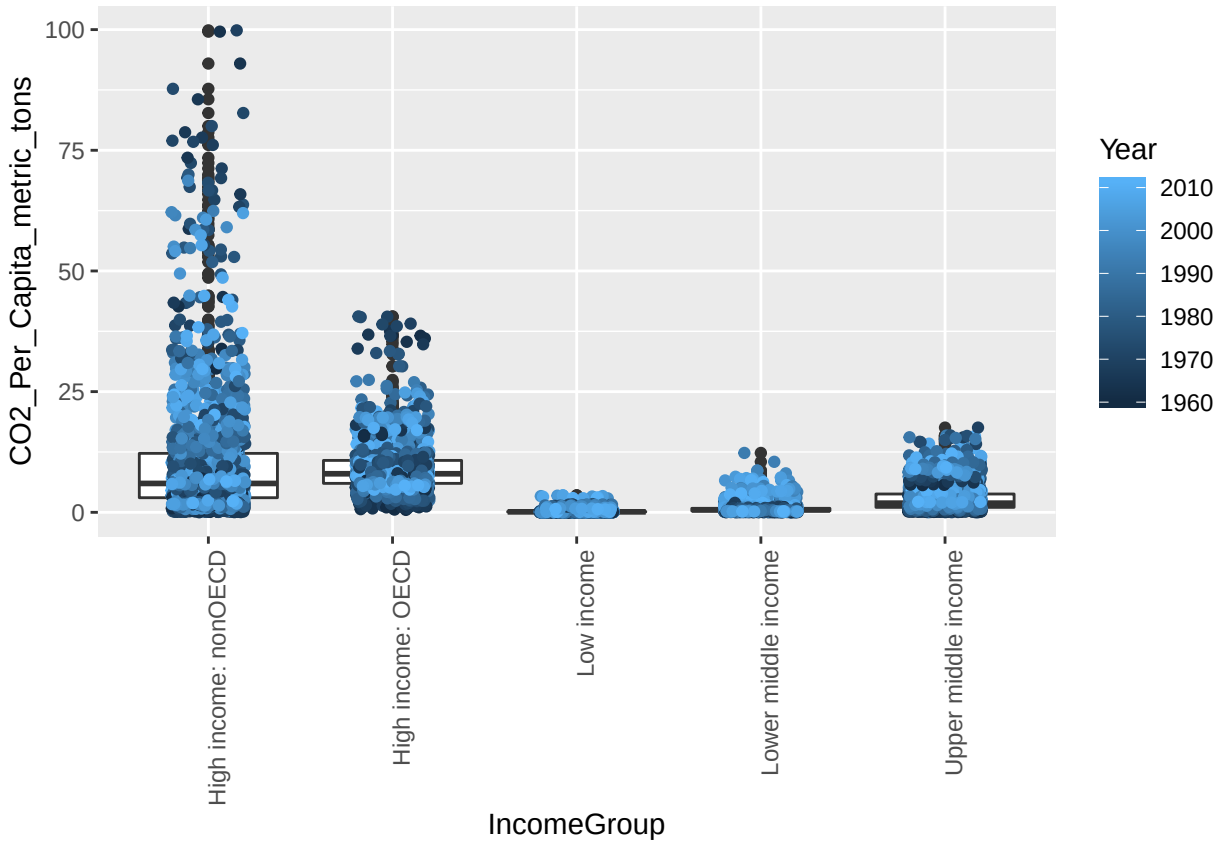
```
## Warning: Removed 2095 rows containing missing values (geom_point).
```




```
data_clean %>% ggplot(aes(IncomeGroup,CO2_Per_Capita_metric_tons,colour=Year)) + geom_boxplot() + theme
```

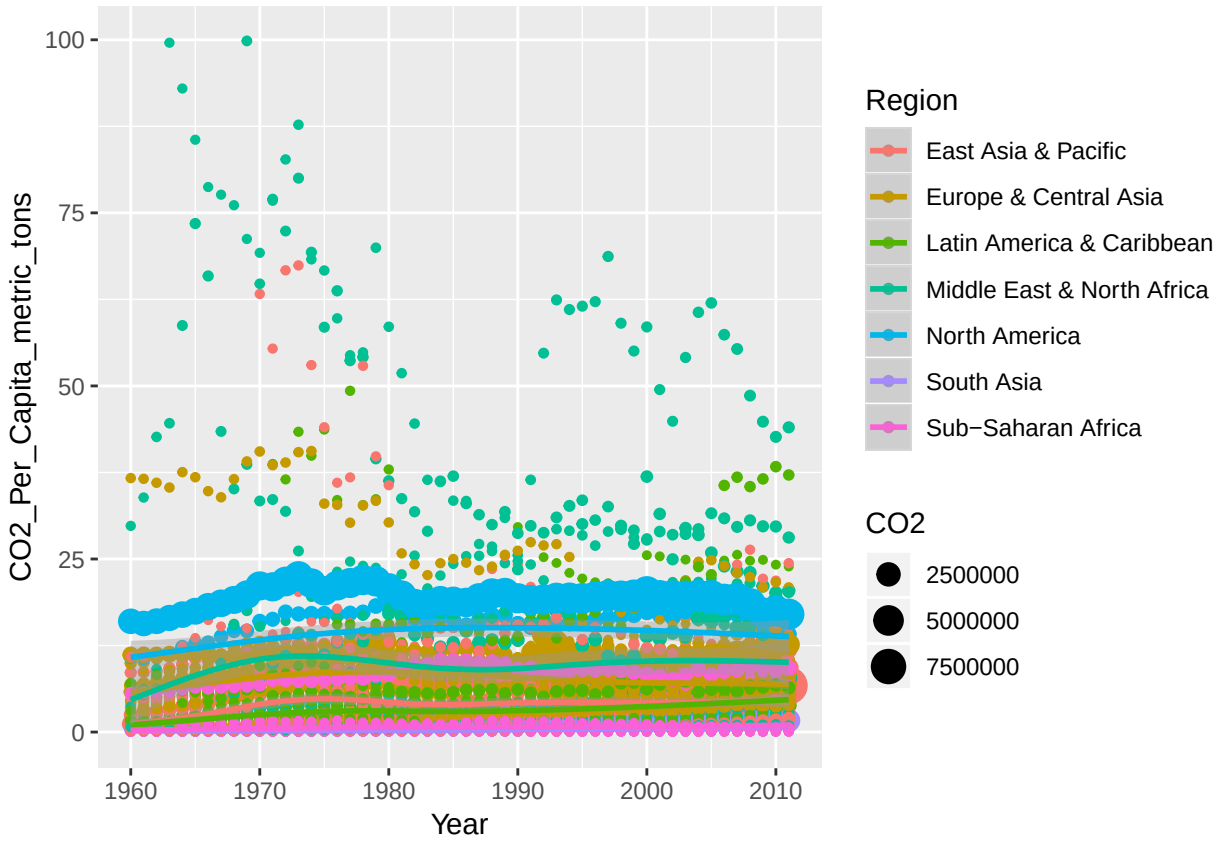
```
## Warning: Removed 2098 rows containing non-finite values (stat_boxplot).
```

```
## Warning: Removed 2098 rows containing missing values (geom_point).
```




```
data_clean %>% ggplot(aes(Year,CO2_Per_Capita_metric_tons,colour=Region)) + geom_point(aes(size=CO2) ) +
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## Warning: Removed 2098 rows containing non-finite values (stat_smooth).
## Warning: Removed 2098 rows containing missing values (geom_point).
```

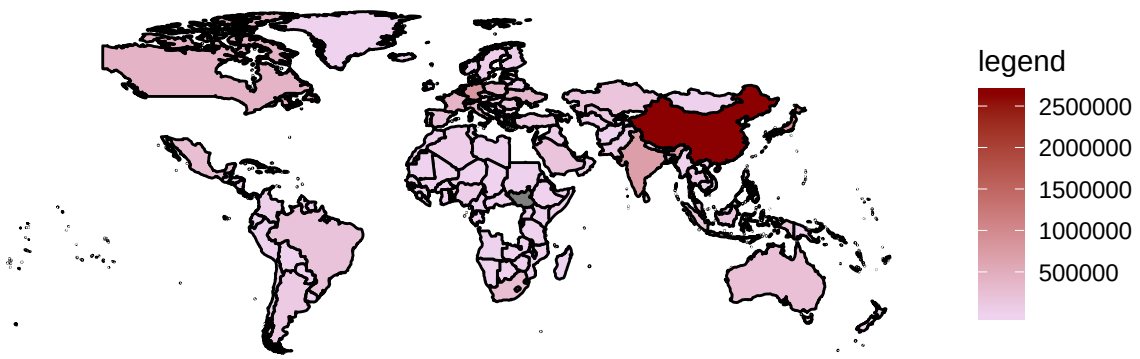



```
WorldData <- map_data('world')
```

```
##
## Attaching package: 'maps'
```

```
## The following object is masked from 'package:purrr':
##
##     map
```

```
#df <-data.frame(region=c('Hungary','Lithuania','Argentina'),value=c(4,10,11))
color= data_clean %>% group_by(Country_Name) %>% summarise(media=mean(CO2,na.rm=TRUE))
Mydata_plot <- inner_join(WorldData , color,by=c("region"="Country_Name"))
Mydata_plot %>% ggplot() +
  geom_polygon(data=Mydata_plot, aes(x=long, y=lat, group = group,fill=media),colour="black") +
  scale_fill_continuous(low = "thistle2", high = "darkred", guide="colorbar") +
  theme_bw() +
  labs(fill = "legend" ,title = "Title", x="", y="") +
  scale_y_continuous(breaks=c()) +
  scale_x_continuous(breaks=c()) +
  theme(panel.border =  element_blank())+coord_fixed(1)
```



7 Análisis de datos 2018/2019: Práctica Final del Bloque 1: Datos de emisiones de CO2 en el mundo.

7.1 Modelo de Datos CO2 y fuente de los datos

El siguiente enlace WorldBankCO2 nos da acceso a un conocido data set de THE WORLD BANK. En concreto la versión de este data set es la de de Tableau Open Data Sets una colección de datos del programa Tableau que es un programa para representar gráficas, paneles de control o *dahsboards* y los llamados KPIs.

7.1.1 Contexto mundial en emisiones de contaminantes

Los siguientes enlaces sirven para saber el contexto de los datos de emisiones mundiales de CO2 . A partir de podéis buscar más.

- Protocolo de kioto (wikipedia)
- Cambio Climático .org
- Acuerdo de París
- Acuerdo de París Comisón Europea

7.2 Cuestiones

Redactar un informe que responda las siguientes cuestiones siguiendo las indicaciones de la cuestión 0.

Hay que subir a la actividad correspondiente de la asignatura en Aula Digital el fichero `.Rmd` y el `.html`.

7.2.1 Cuestión 0

Tenéis que resolver las siguientes cuestiones editando un informe siguiendo las siguientes instrucciones:

- La salida debe tener índice navegable y las `chunks` deben cachear los datos.
- El código debe estar bien indentado, con los comentarios necesarios y los nombres de las variables y funciones suficientemente informativos y en un solo idioma. Las funciones y nombres de variables dentro del texto deben estar en la fuente del código.
- Se debe hacer referencia a la fuente de los datos.
- Cada salida debe tener una conclusión debidamente redactada, sucinta y clara.
- Esta parte se evalúa como presentación global y vale *2.5 puntos*.

7.2.2 Cuestión 1

1. Cargar y depurar la tabla de datos. A partir de las hojas del excel de datos **raw** (`CO2` y `CO2pc`) y de la hoja de metadatos construid con código de `tidyverse` (`dplyr` y compañía) entendible y elegante una tibble que contenga, para cada observación y variable con el tipo de dato adecuado (`numeric`,

`character`, `factor`...) y con el nombre que se pone entre paréntesis, las siguientes variables: (*1.25 puntos*)

- El código de País (`Country_Code`).
- El nombre del País (`Country_Name`).
- El año de la observación (`Year`)
- La observación de CO2 para ese año y país (`CO2`)
- La observación del CO2 per cápita para ese año y país (`CO2pc`).
- Los metadatos de región e ingresos (`Region`, `Income_group`).

2. Mostrad unos resúmenes preliminares de los datos (sin agrupar, agrupados es la siguiente cuestión) y comentad la estructura y una descripción detallada de qué significa cada variable y si es necesario de los niveles o de los valores de cada variable (en especial de los metadatos). (*1.25 puntos*)

7.2.3 Cuestión 2

Se pide:

1. Haced algunas estadísticas de emisiones que muestren las evolución temporal de las emisiones de `CO2` y `CO2pc` a lo largo de los años y agrupadas para cada variables de metadatos. (*1.25 puntos*)
2. Representar con dos o más gráficos de ggplot las series temporales de las gráficas anteriores donde se muestre como `x` el `Year` y como `y` el `CO2` o el `CO2pc` poniendo nombres adecuados y explicativos a los gráficos y a las leyendas. Representad también las variables de metadatos o la `y` ausente (cuando es el `CO2` representad el `CO2pc` y viceversa) (*1.25 puntos*)

7.2.4 Cuestión 3

En la práctica del Taller del CO2 que se adjunta y se explicó en clase) se representó un mapa con alguna de las informaciones que contiene la tabla por país.

Ese mapa tenía algunos problemas:

- Con el código de país o el nombre del país de la tabla de CO2.
- De algunos países no se dibujaba su frontera.

Corregid estos errores con el mapa que se dio en el ejemplo o con otros mapas que podáis utilizar con `ggplot2`.

Se pide:

1. Construid una tibble *adecuada* que contenga los datos del mapa y los datos de la tabla de `CO2`. (*1 punto*)
2. Dibujad un mapa por países coloreando en función de la cantidad `CO2` o por la cantidad `CO2pc` (*1 punto*)
3. Idead y dibujad dos mapas más que incorporen de alguna manera alguno de los metadatos. (*0.5 pt + 2 puntos extra si los mapas son muy informativos*)